

When to Use MLP, CNN, and RNN Neural Networks

What neural network is appropriate for your predictive modeling problem?

It can be difficult for a beginner to the field of deep learning to know what type of network to use. There are so many types of networks to choose from and new methods being published and discussed every day.

To make things worse, most neural networks are flexible enough that they work (make a prediction) even when used with the wrong type of data or prediction problem.

In this post, you will discover the suggested use for the three main classes of artificial neural networks.

After reading this post, you will know:

- Which types of neural networks to focus on when working on a [predictive modeling](#) problem.
- When to use, not use, and possibly try using an MLP, CNN, and RNN on a project.
- To consider the use of hybrid models and to have a clear idea of your project goals before selecting a model.

Discover how to develop deep learning models for a range of predictive modeling problems with just a few lines of code [in my new book](#), with 18 step-by-step tutorials and 9 projects. Let's get started.



When to Use MLP, CNN, and RNN Neural Networks

Photo by [PRODAVID S. FERRY III, DDS](#), some rights reserved.

Overview

This post is divided into five sections; they are:

1. What Neural Networks to Focus on?
2. When to Use Multilayer Perceptrons?
3. When to Use Convolutional Neural Networks?
4. When to Use Recurrent Neural Networks?
5. Hybrid Network Models

What Neural Networks to Focus on?

[Deep learning](#) is the application of artificial neural networks using modern hardware.

It allows the development, training, and use of neural networks that are much larger (more layers) than was previously thought possible.

There are thousands of types of specific neural networks proposed by researchers as modifications or tweaks to existing models. Sometimes wholly new approaches.

As a practitioner, I recommend waiting until a model emerges as generally applicable. It is hard to tease out the signal of what works well generally from the noise of the vast number of publications released daily or weekly.

There are three classes of artificial neural networks that I recommend that you focus on in general. They are:

- Multilayer Perceptrons (MLPs)
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)

These three classes of networks provide a lot of flexibility and have proven themselves over decades to be useful and reliable in a wide range of problems. They also have many subtypes to help specialize them to the quirks of different framings of prediction problems and different datasets.

Now that we know what networks to focus on, let's look at when we can use each class of neural network.

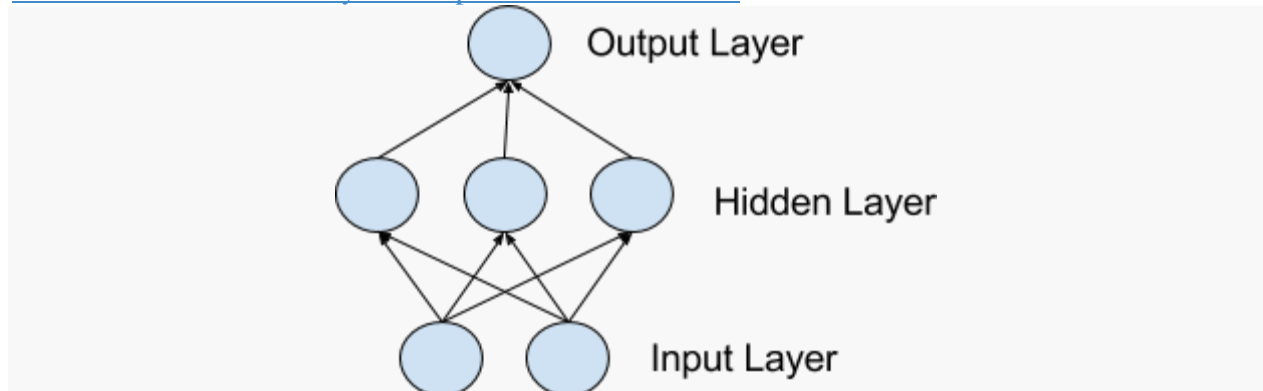
When to Use Multilayer Perceptrons?

Multilayer Perceptrons, or MLPs for short, are the classical type of neural network.

They are comprised of one or more layers of neurons. Data is fed to the input layer, there may be one or more hidden layers providing levels of abstraction, and predictions are made on the output layer, also called the visible layer.

For more details on the MLP, see the post:

- [Crash Course On Multi-Layer Perceptron Neural Networks](#)



Model of a Simple Network

MLPs are suitable for classification prediction problems where inputs are assigned a class or label.

They are also suitable for regression prediction problems where a real-valued quantity is predicted given a set of inputs. Data is often provided in a tabular format, such as you would see in a CSV file or a spreadsheet.

Use MLPs For:

- Tabular datasets
- Classification prediction problems
- Regression prediction problems

They are very flexible and can be used generally to learn a mapping from inputs to outputs.

This flexibility allows them to be applied to other types of data. For example, the pixels of an image can be reduced down to one long row of data and fed into a MLP. The words of a document can also be reduced to one long row of data and fed to a MLP. Even the lag observations for a time series prediction problem can be reduced to a long row of data and fed to a MLP.

As such, if your data is in a form other than a tabular dataset, such as an image, document, or time series, I would recommend at least testing an MLP on your problem. The results can be used as a baseline point of comparison to confirm that other models that may appear better suited add value.

Try MLPs On:

- Image data
- Text Data
- Time series data
- Other types of data

When to Use Convolutional Neural Networks?

Convolutional Neural Networks, or CNNs, were designed to map image data to an output variable.

They have proven so effective that they are the go-to method for any type of prediction problem involving image data as an input.

For more details on CNNs, see the post:

- [Crash Course in Convolutional Neural Networks for Machine Learning](#)

The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. This allows the model to learn position and scale in variant structures in the data, which is important when working with images.

Use CNNs For:

- Image data
- Classification prediction problems
- Regression prediction problems

More generally, CNNs work well with data that has a spatial relationship.

The CNN input is traditionally two-dimensional, a field or matrix, but can also be changed to be one-dimensional, allowing it to develop an internal representation of a one-dimensional sequence.

This allows the CNN to be used more generally on other types of data that has a spatial relationship. For example, there is an order relationship between words in a document of text. There is an ordered relationship in the time steps of a time series.

Although not specifically developed for non-image data, CNNs achieve state-of-the-art results on problems such as document classification used in sentiment analysis and related problems.

Try CNNs On:

- Text data
- Time series data
- Sequence input data

When to Use Recurrent Neural Networks?

Recurrent Neural Networks, or RNNs, were designed to work with sequence prediction problems.

Sequence prediction problems come in many forms and are best described by the types of inputs and outputs supported.

Some examples of sequence prediction problems include:

- **One-to-Many:** An observation as input mapped to a sequence with multiple steps as an output.
- **Many-to-One:** A sequence of multiple steps as input mapped to class or quantity prediction.
- **Many-to-Many:** A sequence of multiple steps as input mapped to a sequence with multiple steps as output.

The Many-to-Many problem is often referred to as sequence-to-sequence, or seq2seq for short.

For more details on the types of sequence prediction problems, see the post:

- [Gentle Introduction to Models for Sequence Prediction with Recurrent Neural Networks](#)

Recurrent neural networks were traditionally difficult to train.

The Long Short-Term Memory, or LSTM, network is perhaps the most successful RNN because it overcomes the problems of training a recurrent network and in turn has been used on a wide range of applications.

For more details on RNNs, see the post:

- [Crash Course in Recurrent Neural Networks for Deep Learning](#)

RNNs in general and LSTMs in particular have received the most success when working with sequences of words and paragraphs, generally called natural language processing.

This includes both sequences of text and sequences of spoken language represented as a time series. They are also used as generative models that require a sequence output, not only with text, but on applications such as generating handwriting.

Use RNNs For:

- Text data
- Speech data
- Classification prediction problems
- Regression prediction problems
- Generative models

Recurrent neural networks are not appropriate for tabular datasets as you would see in a CSV file or spreadsheet. They are also not appropriate for image data input.

Don't Use RNNs For:

- Tabular data
- Image data

RNNs and LSTMs have been tested on time series forecasting problems, but the results have been poor, to say the least. Autoregression methods, even linear methods often perform much better. LSTMs are often outperformed by simple MLPs applied on the same data.

For more on this topic, see the post:

- [On the Suitability of Long Short-Term Memory Networks for Time Series Forecasting](#)

Nevertheless, it remains an active area.

Perhaps Try RNNs on:

- Time series data

Hybrid Network Models

A CNN or RNN model is rarely used alone.

These types of networks are used as layers in a broader model that also has one or more MLP layers. Technically, these are a hybrid type of neural network architecture.

Perhaps the most interesting work comes from the mixing of the different types of networks together into hybrid models.

For example, consider a model that uses a stack of layers with a CNN on the input, LSTM in the middle, and MLP at the output. A model like this can read a sequence of image inputs, such as a video, and generate a prediction. This is called a [CNN LSTM architecture](#).

The network types can also be stacked in specific architectures to unlock new capabilities, such as the reusable image recognition models that use very deep CNN and MLP networks that can be added to a new LSTM model and used for captioning photos. Also, the encoder-decoder LSTM networks that can be used to have input and output sequences of differing lengths.

It is important to think clearly about what you and your stakeholders require from the project first, then seek out a network architecture (or develop one) that meets your specific project needs.

For a good framework to help you think about your data and prediction problems, see the post:

- [How to Define Your Machine Learning Problem](#)

Further Reading

This section provides more resources on the topic if you are looking to go deeper.

- [What Is Deep Learning?](#)
- [Crash Course On Multi-Layer Perceptron Neural Networks](#)
- [Crash Course in Convolutional Neural Networks for Machine Learning](#)
- [Crash Course in Recurrent Neural Networks for Deep Learning](#)

- [Gentle Introduction to Models for Sequence Prediction with Recurrent Neural Networks](#)
- [How to Define Your Machine Learning Problem](#)

Summary

In this post, you discovered the suggested use for the three main classes of artificial neural networks.

Specifically, you learned:

- Which types of neural networks to focus on when working on a predictive modeling problem.
- When to use, not use, and possibly try using an MLP, CNN, and RNN on a project.
- To consider the use of hybrid models and to have a clear idea of your project goals before selecting a model.

Pixel Recurrent Neural Networks

Our method models the discrete probability of the raw pixel values and encodes the **complete** set of dependencies in the image. Architectural novelties include fast two-dimensional recurrent layers and an effective use of residual connections in deep recurrent networks.

1. Image generation model:

Pixel Recurrent Neural Networks

Image Generation Models

-Three image generation approaches are dominating the field:

Variational AutoEncoders (VAE)

Encoder: $q_\phi(z|x)$

Latent variable: $z \sim p_\theta(z)$

Decoder: $x \sim p_\theta(x|z)$

Generative Adversarial Networks (GAN)

Real: x

Fake: \hat{x}

Discriminator: $D \rightarrow \text{Real/Fake?}$

Generator: G (generate)

Latent variable: z

Objective: $\min_G \max_D V(D, G)$

Loss: $= \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))]$

Autoregressive Models

Sequence: x_1, x_2, \dots, x_n

Probability: $p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$

	VAE	GAN	Autoregressive Models
Pros.	- Efficient inference with approximate latent variables.	- generate sharp image. - no need for any Markov chain or approx networks during sampling.	- very simple and stable training process - currently gives the best log likelihood. - tractable likelihood
Cons.	- generated samples tend to be blurry.	- difficult to optimize due to unstable training dynamics.	- relatively inefficient during sampling

(cf. <https://openai.com/blog/generative-models/>)

2. PIXELRNN WORKS REVIEW:

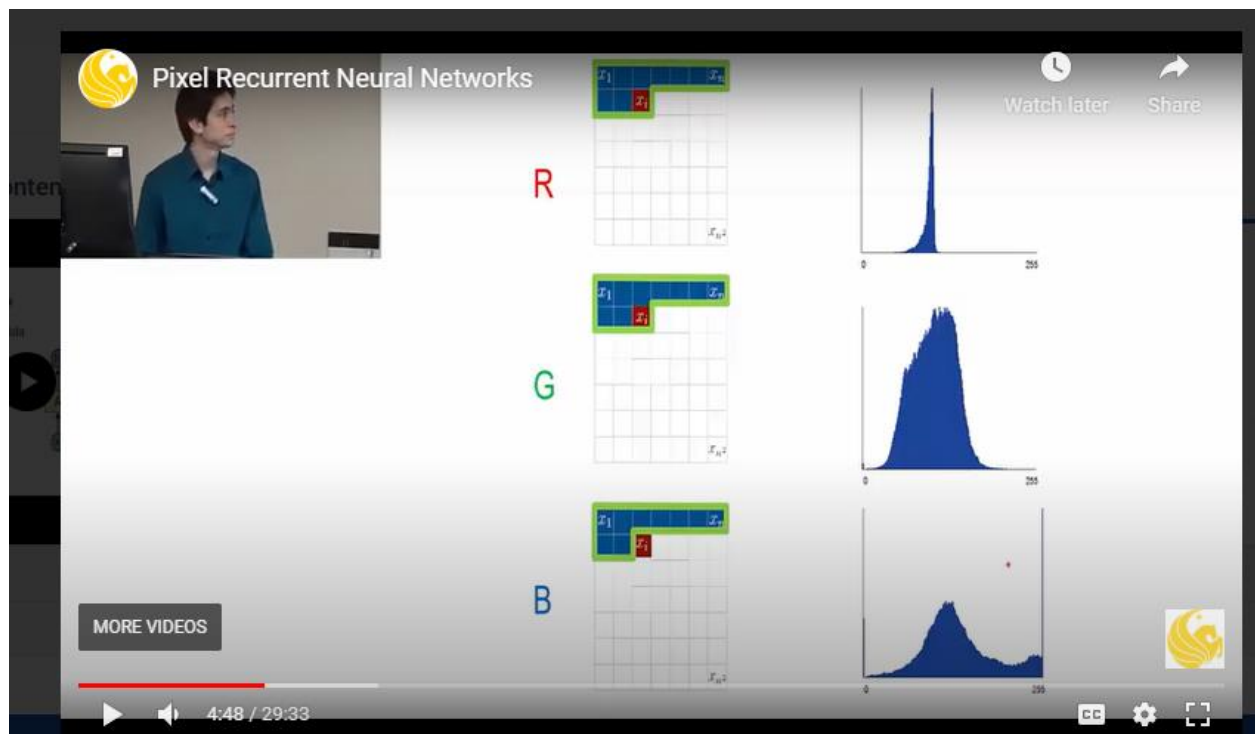
Pixel Recurrent Neural Networks

PixelRNN - Model

- Sequence of pixels
- Pixel is dependent on previous pixels

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$$

3. IMAGE Generation:



4. typical architecture

Pixel Recurrent Neural Networks

Typical Architecture

Image (N, N, 3) → Conv 7x7 → Block → Block ... Block → Conv 1x1 → Conv 1x1 → Soft max → Image (N, N, 3, 256)

MORE VIDEOS

5:27 / 29:33

6. pixel CNN {(generation process)}

Pixel Recurrent Neural Networks

Generated image so far

15 Conv Layers

2 Conv Layers (with ReLU)

Softmax

R

G

B

MORE VIDEOS

6:02 / 29:33

Pixel Recurrent Neural Networks

generated image so far

Conv Layer 1

128 feature maps

MORE VIDEOS

6:23 / 29:33

Watch later Share

Pixel Recurrent Neural Networks

15 Layers

2 Layers (with ReLU)

Softmax


1024 feature maps

256 feature maps

MORE VIDEOS

6:40 / 29:33

Watch later Share

 Pixel Recurrent Neural Networks

15 Layers

2 Layers (with ReLU)

1024 feature maps

256 feature maps

Softmax


Watch later


Share

MORE VIDEOS

7:13 / 29:33

CC



 Pixel Recurrent Neural Networks

15 Layers

2 Layers (with ReLU)

1024 feature maps

256 feature maps

Softmax


Watch later

Share

MORE VIDEOS

7:17 / 29:33

CC



Pixel Recurrent Neural Networks

15 Layers

2 Layers (with ReLU)

Softmax

MORE VIDEOS

7:34 / 29:33

CC

Settings

Full Screen

7. pixelCNN (TRAINING)

Pixel Recurrent Neural Networks

15 Layers

2 Layers (with ReLU)

Softmax

MORE VIDEOS

8:33 / 29:33

8. PixelCNN (advantage and disadvantage)

Pixel Recurrent Neural Networks

PixelCNN Advantages/Disadvantages

- Fastest to train
- Smallest receptive field
- Does not use all available context

MORE VIDEOS

11:04 / 29:33

9. RNN REVIEW

Pixel Recurrent Neural Networks

RNN Review

Sequence of data

Diagram illustrating the unrolled structure of an RNN. The left side shows a single recurrent unit 'A' receiving input x_t and producing output h_t , with a feedback loop. The right side shows the unrolled sequence of units 'A' from $t=0$ to t . Each unit A_t takes input x_t and the previous hidden state h_{t-1} to produce h_t . The inputs $x_0, x_1, x_2, \dots, x_t$ are shown as blue circles, and hidden states $h_0, h_1, h_2, \dots, h_t$ are shown as purple circles.

11:28 / 29:33

10. RNN FOR GENERATION IMAGE

Pixel Recurrent Neural Networks

RNN for Image Generation

Diagram illustrating the RNN for image generation. The left side shows the unrolled sequence of units 'A' from $t=0$ to t . Each unit A_t takes input x_t and the previous hidden state h_{t-1} to produce h_t . The inputs $x_0, x_1, x_2, \dots, x_t$ are shown as blue circles. The hidden states $h_0, h_1, h_2, \dots, h_t$ are shown as purple circles. The outputs $h_0, h_1, h_2, \dots, h_t$ are shown as colored squares. A 3x3 grid of colored squares is shown on the right, representing the generated image.

MORE VIDEOS

12:28 / 29:33

11.LSTM EQUATION

Pixel Recurrent Neural Networks

LSTM Equations

$$i = \sigma(x_i U^i + h_{i-1} W^i)$$

$$f = \sigma(x_i U^f + h_{i-1} W^f)$$

$$o = \sigma(x_i U^o + h_{i-1} W^o)$$

$$g = \tanh(x_i U^g + h_{i-1} W^g)$$

$$c_i = c_{i-1} \circ f + g \circ i$$

$$h_i = \tanh(c_i) \circ o$$

Gates - Control how much information is allowed through

States - Hold information about all time steps up till now $\{0, i, \dots, i-1, i\}$

MORE VIDEOS

13:45 / 29:33

Pixel Recurrent Neural Networks

LSTM Equations

$$i = \sigma(x_i \cancel{U^i} + h_{i-1} \cancel{W^i})$$

$$f = \sigma(x_i \cancel{U^f} + h_{i-1} \cancel{W^f})$$

$$o = \sigma(x_i \cancel{U^o} + h_{i-1} \cancel{W^o})$$

$$g = \tanh(x_i \cancel{U^g} + h_{i-1} \cancel{W^g})$$

$$c_i = c_{i-1} \circ f + g \circ i$$

$$h_i = \tanh(c_i) \circ o$$

Like Convolutional LSTM - replaced fully-connected layer with convolutional layer

$$[o_i, f_i, i_i, g_i] = \sigma(\underline{K^{ss} \otimes h_{i-1}} + \underline{K^{is} \otimes x_i})$$

$$c_i = f_i \odot c_{i-1} + i_i \odot g_i$$

$$h_i = o_i \odot \tanh(c_i)$$

MORE VIDEOS

14:23 / 29:33

12.ROW LSTM



Pixel Recurrent Neural Networks



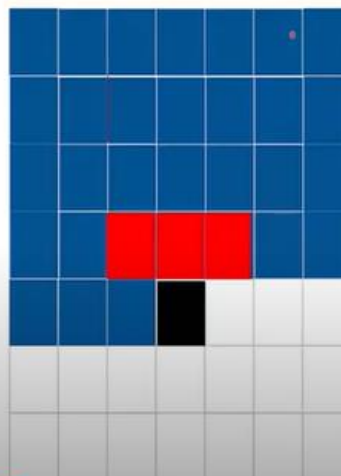
Watch later



Share



Row LSTM



MORE VIDEOS



14:57 / 29:33

