Four Common Types of Neural Network Layers

(and when to use them)

Martin Isaksson Jun 6, 2020 · 4 min read ★

Neural networks (NN) are the backbone of many of today's machine learning (ML) models, loosely mimicking the neurons of the human brain to recognize patterns from input data. As a result, numerous types of neural network topologies have been designed over the years, built using different types of neural network layers.

And with today's vast array of ML frameworks and tools, just about anyone with a bit of ML knowledge can easily build a model with different types of neural network topologies. For the most part, it's all about knowing what problems each type of neural network excels at solving, and optimizing their hyperparameter configurations.

The four most common types of neural network layers are *Fully connected*, *Convolution*, *Deconvolution*, and *Recurrent,* and below you will find what they are and how they can be used.

Fully Connected Layer

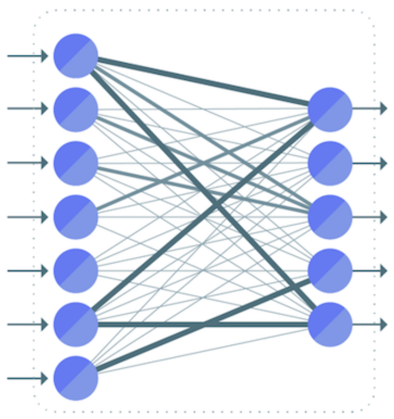

Image by PerceptiLabs

Fully connected layers connect every neuron in one layer to every neuron in the next layer. Fully connected layers are found in all different types of neural networks ranging from standard neural networks to convolutional neural networks (CNN).

Fully connected layers can become computationally expensive as their input grows, resulting in a combinatorial explosion of vector operations to be performed, and potentially poor scalability. As such, they are commonly used for specific purposes within neural networks such as classifying image data.

Use Cases

- Experimentation or learning ML using fully connected neural networks.
- In CNNs to classify images for computer vision.

Hyperparameters Commonly Associated with Fully Connected Layers

- Activation function
- Number of neurons
- Dropout

Convolution Layer

Image by PerceptiLabs

A Convolution Layer is an important type of layer in a CNN. Its most common use is for detecting features in images, in which it uses a *filter* to scan an image, a few pixels at a time, and outputs a *feature map* that classifies each feature found.

The filter (sometimes called *kernel*) is a set of n-dimensional weights that are multiplied against the input, where the filter's dimensions match that of the input (e.g., two dimensions when dealing with 2D images). The filter describes the probabilities that a given pattern of pixels represents a feature. Thus the number of filter weights (i.e., size of the filter) are smaller than the input, and the multiplication performed by the layer's convolution process is performed on "patches" of the image that match the filter size.

Multiplication is systematically repeated from left to right and top to bottom over the entire image to detect features. The number of pixels by which the filter moves for the next iteration is called the *stride*. *Padding* may be added around the input image to ensure that the filter always fits within the total bounds of the image for a given stride.

Use Cases

- Analyzing imagery for image recognition and classification.

Hyperparameters Commonly Associated with Convolution Layer

- Dimensionality
- Patch size
- Stride
- Number of feature maps to generate
- Padding strategy
- Activate function
- Dropout
- Pooling

Deconvolution Layer

Image by PerceptiLabs

A Deconvolution Layer is a transposed convolution process that effectively upsamples data to a higher resolution. This can include image data and/or feature maps generated from a convolution layer, or other types of data. For image data, the upsampling resolution output by deconvolution may be the same as the original input image, or may be different.

Common Use Cases

- Upsampling images

Hyperparameters Commonly Associated with Deconvolution Layer

- Dimensionality
- Stride
- Number of feature maps to generate
- Padding strategy
- Activate function
- Dropout

Recurrent Layer

Image by PerceptiLabs

A Recurrent Layer includes a “looping” capability such that its input consists of both the data to analyze as well as the output from a previous calculation performed by that layer. Recurrent layers form the basis of recurrent neural networks (RNNs), effectively providing them with *memory* (i.e., maintain a state across iterations), while their recursive nature makes RNNs useful for cases involving sequential data like natural language and time series. They’re also useful for mapping inputs to outputs of different types and dimensions.

Common Use Cases

- Classifying sentiments into positive and negative.
- Generating text descriptions of what an image contains.
- Translating paragraphs of text to another language.

Hyperparameters Commonly Associated with Recurrent Layer

- Dimensionality
- Type of recurrent neural network (LSTM, GRU, or standard RNN layer)
- Return sequence
- Dropout

Conclusion

Neural networks are the current state-of-the-art when it comes to machine learning and there are many topologies and layer types to choose from. Each type of neural network excels at solving a specific domain of problems, and each is tuned with hyperparameters that optimize those solutions. Furthermore, ML practitioners can now access many ML frameworks and tools that make it easier than ever to implement ML models built around neural network topologies.

For more information, check out my Machine Learning Handbook that provides additional detail about neural networks and other aspects of ML.

Machine Learning · Neural Networks · Hyperparameter Tuning · Ml Framework · Machine Learning Tools