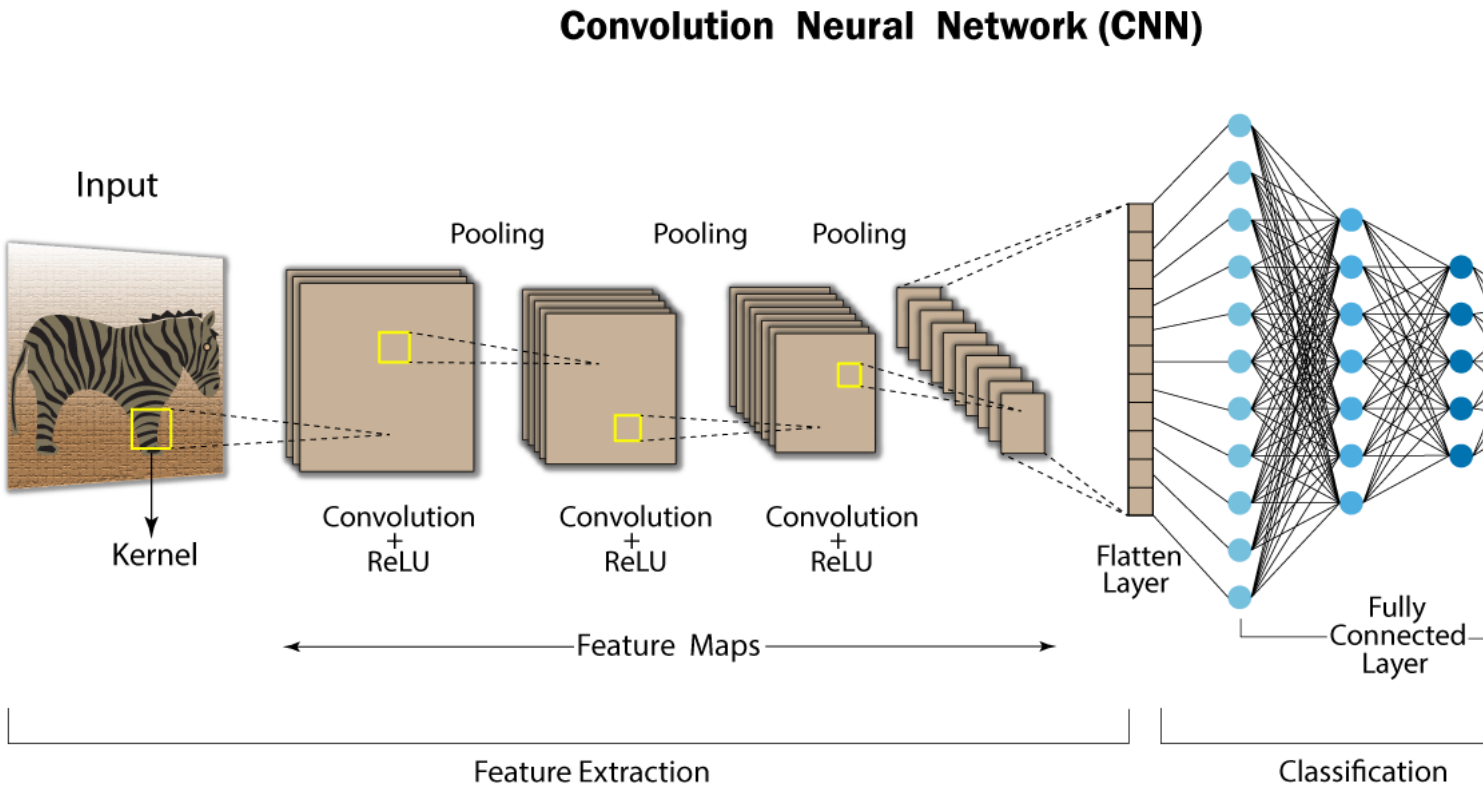
Basics of CNN in Deep Learning

BEGINNER COMPUTER VISION DEEP LEARNING IMAGE PYTHON

This article was published as a part of the Data Science Blogathon.

What is Convolutional Neural Network?

Convolutional Neural Networks also known as CNNs or ConvNets, are a type of feed-forward artificial neural network whose connectivity structure is inspired by the organization of the animal visual cortex. Small clusters of cells in the visual cortex are sensitive to certain areas of the visual field. Individual neuronal cells in the brain respond or fire only when certain orientations of edges are present. Some neurons activate when shown vertical edges, while others fire when shown horizontal or diagonal edges. A convolutional neural network is a type of artificial neural network used in deep learning to evaluate visual information. These networks can handle a wide range of tasks involving images, sounds, texts, videos, and other media. Professor Yann LeCunn of Bell Labs created the first successful convolution networks in the late 1990s.



Source: Medium.com

Convolutional Neural Networks (CNNs) have an input layer, an output layer, numerous hidden layers, and millions of parameters, allowing them to learn complicated objects and patterns. It uses convolution and pooling processes to sub-sample the given input before applying an activation function, where all of them

are hidden layers that are partially connected, with the completely connected layer at the end resulting in the output layer. The output shape is similar to the size of the input image.

Convolution is the process of combining two functions to produce the output of the other function. The input image is convoluted with the application of filters in CNNs, resulting in a Feature map. Filters are weights and biases that are randomly generated vectors in the network. Instead of having individual weights and biases for each neuron, CNN uses the same weights and biases for all neurons. Many filters can be created, each of which catches a different aspect from the input. Kernels are another name for filters.

Convolutional Layer

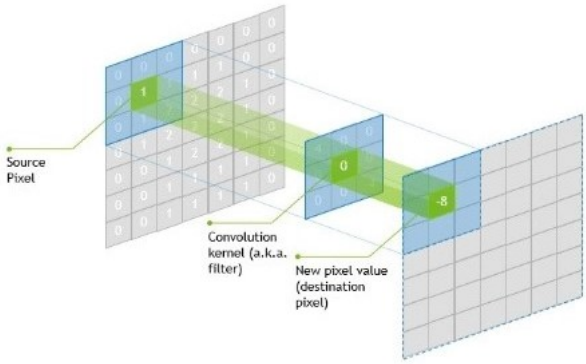
In convolutional neural networks, the major building elements are convolutional layers. This layer often contains input vectors, such as an image, filters, such as a feature detector, and output vectors, such as a feature map. The image is abstracted to a feature map, also known as an activation map, after passing through a convolutional layer.

Feature Map = Input Image x Feature Detector

The input is convolved by convolutional layers, which then pass the output to the next layer. This is analogous to a neuron's response to a single stimulus in the visual cortex. Each convolutional neuron only processes data for the receptive field it is assigned to.

A convolution is a grouping function in mathematics. Convolution occurs in CNNs when two matrices (rectangular arrays of numbers arranged in columns and rows) are combined to generate a third matrix.

In the convolutional layers of a CNN, these convolutions are used to filter input data and find information.



Source: Cadalyst.com

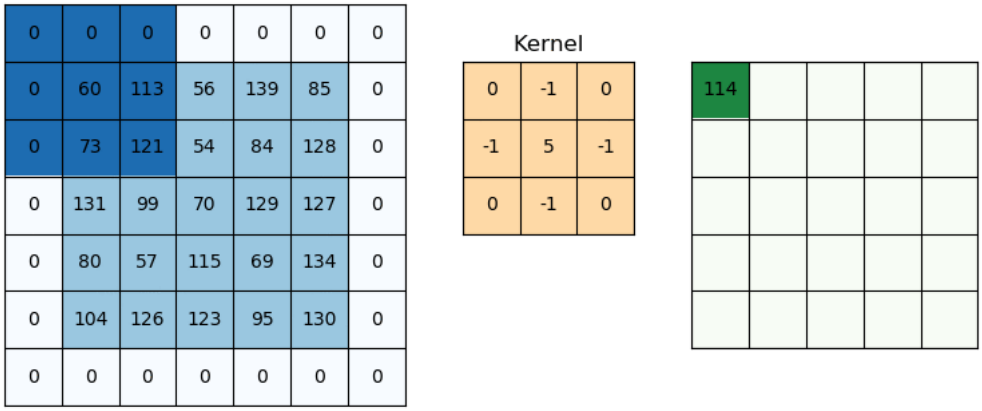
The kernel's centre element is put above the source pixel. After that, the source pixel is replaced with a weighted sum of itself and neighboring pixels.

Parameter sharing and local connectivity are two principles used in CNNs. All neurons in a feature map share weights, which is known as parameter sharing. Local connection refers to the idea of each neural being connected to only a part of the input image (as opposed to a neural network in which all neurons are fully connected). This reduces the number of parameters in the system and speeds up the calculation.

Padding and Stride

Padding and stride have an impact on how the convolution procedure is carried out. Padding and stride can be used to increase or decrease the dimensions (height and width) of input/output vectors.

Padding is a term used in convolutional neural networks to describe how many pixels are added to an image when it is processed by the CNN kernel. If the padding in a CNN is set to zero, for example, every pixel value-added will have the value zero. If the zero padding is set to one, a one-pixel border with a pixel value of zero will be added to the image.



Source: vitalflux.com

Padding works by increasing the processing region of a convolutional neural network. The kernel is a neural network filter that moves through a picture, scanning each pixel and turning the data into a smaller or bigger format. Padding is added to the image frame to aid the kernel in processing the image by providing more room for the kernel to cover the image. Adding padding to a CNN-processed image provides for more accurate image analysis.

Source: Computer.org

Stride determines how the filter convolves over the input matrix, i.e. how many pixels shift. When stride is set to 1, the filter moves across one pixel at a time, and when the stride is set to 2, the filter moves across two pixels at a time. The smaller the stride value, the smaller the output, and vice versa.

Pooling

Its purpose is to gradually shrink the representation's spatial size to reduce the number of parameters and computations in the network. The pooling layer treats each feature map separately.

Source: Springer.com

The following are some methods for pooling:

- ***Max-pooling***: It chooses the most significant element from the feature map. The feature map's significant features are stored in the resulting max-pooled layer. It is the most popular method since it produces the best outcomes.

- ***Average pooling***: It entails calculating the average for each region of the feature map.

Pooling gradually reduces the spatial dimension of the representation to reduce the number of parameters and computations in the network, as well as to prevent overfitting. If there is no pooling, the output has the same resolution as the input.

ReLU

The ***rectified linear activation function***, or ReLU for short, is a piecewise linear function that, if the input is positive, outputs the input directly; else, it outputs zero. Because a model that utilizes it is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks.

Source: Superdatascience.com

At the end of CNN, there is a ***Fully connected*** layer of neurons. As in conventional Neural Networks, neurons in a fully connected layer have full connections to all activations in the previous layer and work similarly. After training, the feature vector from the fully connected layer is used to classify images into distinct categories. Every activation unit in the next layer is coupled to all of the inputs from this layer. Overfitting occurs because all of the parameters are occupied in the fully-connected layer. Overfitting can be reduced using a variety of strategies, including dropout.

Soft-max is an activation layer that is typically applied to the network's last layer, which serves as a classifier. This layer is responsible for categorizing provided input into distinct types. A network's non-normalized output is mapped to a probability distribution using the softmax function.

Basic Python Implementation

Importing Some Relevant Libraries

```
import NumPy as np %matplotlib inline import matplotlib.image as mpimg import matplotlib.pyplot as plt import TensorFlow as tf tf.compat.v1.set_random_seed(2019)
```

Loading the MNIST Dataset

```
(X_train,Y_train),(X_test,Y_test) = keras.datasets.mnist.load_data()
```

Scaling our Data

```
X_train = X_train / 255 X_test = X_test / 255
```

#flatenning
X_train_flattened = X_train.reshape(len(X_train), 28*28)
X_test_flattened = X_test.reshape(len(X_test), 28*28)

Designing Neural Network

```
model = keras.Sequential([ keras.layers.Dense(10, input_shape=(784,), activation='sigmoid') ])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train_flattened, Y_train, epochs=5)
```

Output

```
Epoch 1/5 1875/1875 [==============================] - 8s 4ms/step - loss: 0.7187 - accuracy: 0.8141 Epoch 2/5 1875/1875 [==============================] - 6s 3ms/step - loss: 0.3122 - accuracy: 0.9128 Epoch 3/5 1875/1875 [==============================] - 6s 3ms/step - loss: 0.2908 - accuracy: 0.9187 Epoch 4/5 1875/1875 [==============================] - 6s 3ms/step - loss: 0.2783 - accuracy: 0.9229 Epoch 5/5 1875/1875 [==============================] - 6s 3ms/step - loss: 0.2643 - accuracy: 0.9262
```

Conclusion

The goal of this article was to provide an overview of convolutional neural networks and their main applications. These networks, in general, produce excellent classification and recognition results. They're also used to decode audio, text, and video. If the task at hand is to find a pattern in a series, convolutional networks are an excellent choice.

Read more articles about CNNs here.

Article Url - https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/

Debasish Kalita