**How Do Convolutional Layers Work in Deep Learning Neural Networks?**

<https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>

uvod

Convolutional layers are the major building blocks used in convolutional neural networks.

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

In this tutorial, you will discover how convolutions work in the convolutional neural network.

Ovo počet ka 'cilj ovog poglavlja je'…

In this tutorial, you will discover how convolutions work in the convolutional neural network.

After completing this tutorial, you will know:

* Convolutional neural networks apply a filter to an input to create a feature map that summarizes the presence of detected features in the input.
* Filters can be handcrafted, such as line detectors, but the innovation of convolutional neural networks is to learn the filters during training in the context of a specific prediction problem.
* How to calculate the feature map for one- and two-dimensional convolutional layers in a convolutional neural network.

This tutorial is divided into four parts; they are:

1. Convolution in Convolutional Neural Networks
2. Convolution in Computer Vision
3. Power of Learned Filters
4. Worked Example of Convolutional Layers

## CONVOLUTION IN CONVOLUTIONAL NEURAL NETWORKS

The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data.

Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a “*convolution*“.

'težine' u neuralnoj mreži su zapravo težine u filterima?

In the context of a convolutional neural network, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A [dot product](https://en.wikipedia.org/wiki/Dot_product) is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the “scalar product“.

Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

Korištenje istog filtera preko cijele 'slike' omogućava filteru da prepozna određen 'feature' bilo gdje u slici

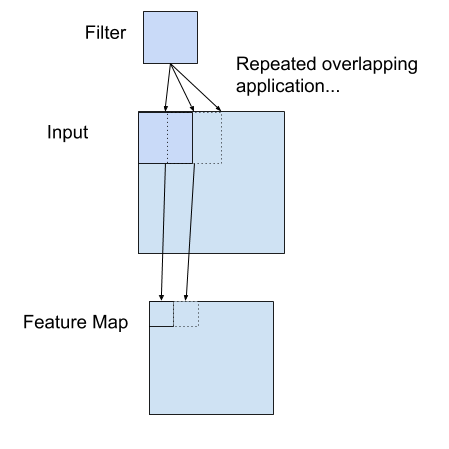
This systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance, e.g. the general interest in whether the feature is present rather than where it was present.

*Invariance to local translation can be a very useful property if we care more about whether some feature is present than exactly where it is. For example, when determining whether an image contains a face, we need not know the location of the eyes with pixel-perfect accuracy, we just need to know that there is an eye on the left side of the face and an eye on the right side of the face.*

— Page 342, [Deep Learning](https://amzn.to/2Dl124s), 2016.

The output from multiplying the filter with the input array one time is a single value. As the filter is applied multiple times to the input array, the result is a two-dimensional array of output values that represent a filtering of the input. As such, the two-dimensional output array from this operation is called a “*feature map*“.

Once a feature map is created, we can pass each value in the feature map through a nonlinearity, such as a ReLU, much like we do for the outputs of a fully connected layer.



Example of a Filter Applied to a Two-Dimensional Input to Create a Feature Map

## CONVOLUTION IN COMPUTER VISION

The idea of applying the convolutional operation to image data is not new or unique to convolutional neural networks; it is a common technique used in computer vision.

Povijest filtera koji se danas koriste u konv mrežama

Historically, filters were designed by hand by computer vision experts, which were then applied to an image to result in a feature map or output from applying the filter then makes the analysis of the image easier in some way.

For example, below is a hand crafted 3×3 element filter for detecting vertical lines:

0.0, 1.0, 0.0

0.0, 1.0, 0.0

0.0, 1.0, 0.0

Applying this filter to an image will result in a feature map that only contains vertical lines. It is a vertical line detector.

You can see this from the weight values in the filter; any pixels values in the center vertical line will be positively activated and any on either side will be negatively activated. Dragging this filter systematically across pixel values in an image can only highlight vertical line pixels.

A horizontal line detector could also be created and also applied to the image, for example:

0.0, 0.0, 0.0

1.0, 1.0, 1.0

0.0, 0.0, 0.0

Combining the results from both filters, e.g. combining both feature maps, will result in all of the lines in an image being highlighted.

A suite of tens or even hundreds of other small filters can be designed to detect other features in the image.

The innovation of using the convolution operation in a neural network is that the values of the filter are weights to be learned during the training of the network.

The network will learn what types of features to extract from the input. Specifically, training under stochastic gradient descent, the network is forced to learn to extract features from the image that minimize the loss for the specific task the network is being trained to solve, e.g. extract features that are the most useful for classifying images as dogs or cats.

## POWER OF LEARNED FILTERS

Learning a single filter specific to a machine learning task is a powerful technique.

Yet, convolutional neural networks achieve much more in practice.

### MULTIPLE FILTERS

Convolutional neural networks do not learn a single filter; they, in fact, learn multiple features in parallel for a given input.

For example, it is common for a convolutional layer to learn from 32 to 512 filters in parallel for a given input.

This gives the model 32, or even 512, different ways of extracting features from an input, or many different ways of both “*learning to see*” and after training, many different ways of “*seeing*” the input data.

This diversity allows specialization, e.g. not just lines, but the specific lines seen in your specific training data.

### MULTIPLE CHANNELS

Color images have multiple channels, typically one for each color channel, such as red, green, and blue.

From a data perspective, that means that a single image provided as input to the model is, in fact, three images.

A filter must always have the same number of channels as the input, often referred to as “*depth*“. If an input image has 3 channels (e.g. a depth of 3), then a filter applied to that image must also have 3 channels (e.g. a depth of 3). In this case, a 3×3 filter would in fact be 3x3x3 or [3, 3, 3] for rows, columns, and depth. Regardless of the depth of the input and depth of the filter, the filter is applied to the input using a dot product operation which results in a single value.

This means that if a convolutional layer has 32 filters, these 32 filters are not just two-dimensional for the two-dimensional image input, but are also three-dimensional, having specific filter weights for each of the three channels. Yet, each filter results in a single feature map. Which means that the depth of the output of applying the convolutional layer with 32 filters is 32 for the 32 feature maps created.

### MULTIPLE LAYERS

Convolutional layers are not only applied to input data, e.g. raw pixel values, but they can also be applied to the output of other layers.

The stacking of convolutional layers allows a hierarchical decomposition of the input.

Consider that the filters that operate directly on the raw pixel values will learn to extract low-level features, such as lines.

The filters that operate on the output of the first line layers may extract features that are combinations of lower-level features, such as features that comprise multiple lines to express shapes.

This process continues until very deep layers are extracting faces, animals, houses, and so on.

This is exactly what we see in practice. The abstraction of features to high and higher orders as the depth of the network is increased.

DALJE NE MORAŠ PREVODIT

PRIMJERI U PRAKSI

## WORKED EXAMPLE OF CONVOLUTIONAL LAYERS

In this section, we’ll look at both a one-dimensional convolutional layer and a two-dimensional convolutional layer example to both make the convolution operation concrete and provide a worked example of using the Keras layers.

### EXAMPLE OF 1D CONVOLUTIONAL LAYERS

We can define a one-dimensional input that has eight elements all with the value of 0.0, with a two element bump in the middle with the values 1.0.

[0, 0, 0, 1, 1, 0, 0, 0]

The input to Keras must be three dimensional for a 1D convolutional layer.

Prva dimenzija je broj uzoraka, druga je duljina svakog uzorka i treća je nroj kanala, jedan kanal za 1D

The first dimension refers to each input sample; in this case, we only have one sample. The second dimension refers to the length of each sample; in this case, the length is eight. The third dimension refers to the number of channels in each sample; in this case, we only have a single channel.

Therefore, the shape of the input array will be [1, 8, 1].

# define input data

data = asarray([0, 0, 0, 1, 1, 0, 0, 0])

data = data.reshape(1, 8, 1)

We will define a model that expects input samples to have the shape [8, 1].

### Example of 2D Convolutional Layer

We can expand the bump detection example in the previous section to a vertical line detector in a two-dimensional image.

Again, we can constrain the input, in this case to a square 8×8 pixel input image with a single channel (e.g. grayscale) with a single vertical line in the middle.

[0, 0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0, 0, 0]

The input to a Conv2D layer must be four-dimensional.

The first dimension defines the samples; in this case, there is only a single sample. The second dimension defines the number of rows; in this case, eight. The third dimension defines the number of columns, again eight in this case, and finally the number of channels, which is one in this case.

Therefore, the input must have the four-dimensional shape [samples, rows, columns, channels] or [1, 8, 8, 1] in this case.