# CRASH COURSE IN CNN FOR ML

<https://machinelearningmastery.com/crash-course-convolutional-neural-networks/>

primjer kako započet poglavlje o konvolucijskim mrežama

In this post you will discover Convolutional Neural Networks for deep learning, also called ConvNets or CNNs. After completing this crash course you will know:

* The building blocks used in CNNs such as convolutional layers and pool layers.
* How the building blocks fit together with a short worked example.
* Best practices for configuring CNNs on your own object recognition tasks.
* References for state of the art networks applied to complex machine learning problems.

## THE CASE FOR COVOLUTIONAL NEURAL NETWORKS

Usporedba obične i konvolucijske NN

Given a dataset of gray scale images with the standardized size of 32×32 pixels each, a traditional feedforward neural network would require 1024 input weights (plus one bias).

This is fair enough, but the flattening of the image matrix of pixels to a long vector of pixel values loses all of the spatial structure in the image. Unless all of the images are perfectly resized, the neural network will have great difficulty with the problem.

CNN ostavlja sačuvanom prostornu vezu među pikselima

Convolutional Neural Networks expect and preserve the spatial relationship between pixels by learning internal feature representations using small squares of input data. Feature are learned and used across the whole image, allowing for the objects in the images to be shifted or translated in the scene and still detectable by the network.

It is this reason why the network is so useful for object recognition in photographs, picking out digits, faces, objects and so on with varying orientation.

In summary, below are some benefits of using convolutional neural networks:

* They use fewer parameters (weights) to learn than a fully connected network.
* They are designed to be invariant to object position and distortion in the scene.
* They automatically learn and generalize features from the input domain.

## BUILDING BLOCKS OF CONVOLUTIONAL NEURAL NETWORKS

There are three types of layers in a Convolutional Neural Network:

1. Convolutional Layers.
2. Pooling Layers.
3. Fully-Connected Layers.

### CONVOLUTIONAL LAYERS

[Convolutional layers](https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/) are comprised of filters and feature maps.

**FILTERS**

The filters are the “neurons” of the layer. They have input weights and output a value. The input size is a fixed square called a patch or a receptive field.

If the convolutional layer is an input layer, then the input patch will be pixel values. If the deeper in the network architecture, then the convolutional layer will take input from a feature map from the previous layer.

**FEATURE MAPS**

The feature map is the output of one filter applied to the previous layer.

A given filter is drawn across the entire previous layer, moved one pixel at a time. Each position results in an activation of the neuron and the output is collected in the feature map. You can see that if the receptive field is moved one pixel from activation to activation, then the field will overlap with the previous activation by (field width – 1) input values.

**ZERO PADDING**

The distance that filter is moved across the input from the previous layers each activation is referred to as the [stride](https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/).

If the size of the previous layer is not cleanly divisible by the size of the filters receptive field and the size of the stride then it is possible for the receptive field to attempt to read off the edge of the input feature map. In this case, techniques like zero padding can be used to invent mock inputs for the receptive field to read.

## POOLING LAYERS

The [pooling layers](https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/) down-sample the previous layers feature map.

Pooling layers follow a sequence of one or more convolutional layers and are intended to consolidate the features learned and expressed in the previous layers feature map. As such, pooling may be consider a technique to compress or generalize feature representations and generally reduce the overfitting of the training data by the model.

They too have a receptive field, often much smaller than the convolutional layer. Also, the stride or number of inputs that the receptive field is moved for each activation is often equal to the size of the receptive field to avoid any overlap.

Pooling layers are often very simple, taking the average or the maximum of the input value in order to create its own feature map.

## FULLY CONNECTED LAYERS

Fully connected layers are the normal flat feed-forward neural network layer.

These layers may have a non-linear activation function or a softmax activation in order to output probabilities of class predictions.

Fully connected layers are used at the end of the network after feature extraction and consolidation has been performed by the convolutional and pooling layers. They are used to create final non-linear combinations of features and for making predictions by the network.

## WORKED EXAMPLE OF A CONVOLUTIONAL NEURAL NETWORK

Let’s make this more concrete by working through how these three layers may be connected together.

### IMAGE INPUT DATA

Let’s assume we have a dataset of grayscale images. Each image has the same size of 32 pixels wide and 32 pixels high, and pixel values are between 0 and 255, g.e. a matrix of 32x32x1 or 1024 pixel values.

Image input data is expressed as a 3-dimensional matrix of width \* height \* channels. If we were using color images in our example, we would have 3 channels for the red, green and blue pixel values, e.g. 32x32x3.

### CONVOLUTIONAL LAYER

We define a convolutional layer with 10 filters and a receptive field 5 pixels wide and 5 pixels high and a stride length of 1.

Because each filter can only get input from (i.e. “see”) 5×5 (25) pixels at a time, we can calculate that each will require 25 + 1 input weights (plus 1 for the bias input).

Dragging the 5×5 receptive field across the input image data with a stride width of 1 will result in a feature map of 28×28 output values or 784 distinct activations per image.

We have 10 filters, so that is 10 different 28×28 feature maps or 7,840 outputs that will be created for one image.

Finally, we know we have 26 inputs per filter, 10 filters and 28×28 output values to calculate per filter, therefore we have a total of 26x10x28x28 or 203,840 “connections” in our convolutional layer, we we want to phrase it using traditional neural network nomenclature.

Convolutional layers also make use of a nonlinear transfer function as part of activation and the rectifier activation function is the popular default to use.

### POOL LAYER

We define a pooling layer with a receptive field with a width of 2 inputs and a height of 2 inputs. We also use a stride of 2 to ensure that there is no overlap.

This results in feature maps that are one half the size of the input feature maps. From 10 different 28×28 feature maps as input to 10 different 14×14 feature maps as output.

We will use a max() operation for each receptive field so that the activation is the maximum input value.

### FULLY CONNECTED LAYER

Finally, we can flatten out the square feature maps into a traditional flat fully connected layer.

We can define the fully connected layer with 200 hidden neurons, each with 10x14x14 input connections, or 1960 + 1 weights per neuron. That is a total of 392,200 connections and weights to learn in this layer.

We can use a sigmoid or softmax transfer function to output probabilities of class values directly.

## CONVOLUTIONAL NEURAL NETWORK BEST PRACTICES

Now that we know about the building blocks for a convolutional neural network and how the layers hang together, we can review some best practices to consider when applying them.

* **Input Receptive Field Dimensions**: The default is 2D for images, but could be 1D such as for words in a sentence or 3D for video that adds a time dimension.
* **Receptive Field Size**: The patch should be as small as possible, but large enough to “see” features in the input data. It is common to use 3×3 on small images and 5×5 or 7×7 and more on larger image sizes.
* **Stride Width**: Use the default stride of 1. It is easy to understand and you don’t need padding to handle the receptive field falling off the edge of your images. This could increased to 2 or larger for larger images.
* **Number of Filters**: Filters are the feature detectors. Generally fewer filters are used at the input layer and increasingly more filters used at deeper layers.
* **Padding**: Set to zero and called zero padding when reading non-input data. This is useful when you cannot or do not want to standardize input image sizes or when you want to use receptive field and stride sizes that do not neatly divide up the input image size.
* **Pooling**: Pooling is a destructive or generalization process to reduce overfitting. Receptive field is almost always set to to 2×2 with a stride of 2 to discard 75% of the activations from the output of the previous layer.
* **Data Preparation**: Consider standardizing input data, both the dimensions of the images and pixel values.
* **Pattern Architecture**: It is common to pattern the layers in your network architecture. This might be one, two or some number of convolutional layers followed by a pooling layer. This structure can then be repeated one or more times. Finally, fully connected layers are often only used at the output end and may be stacked one, two or more deep.
* **Dropout**: CNNs have a habit of overfitting, even with pooling layers. Dropout should be used such as between fully connected layers and perhaps after pooling layers.

## SUMMARY

In this post you discovered convolutional neural networks. You learned about:

* Why CNNs are needed to preserve spatial structure in your input data and the benefits they provide.
* The building blocks of CNN including convolutional, pooling and fully connected layers.
* How the layers in a CNN hang together.
* Best practices when applying CNN to your own problems.

Do you have any questions about convolutional neural networks or about this post? Ask your questions in the comments and I will do my best to answer.