A Gentle Introduction to Padding and Stride for Convolutional Neural Networks

<https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/>

pomišat s poglavljem iz deeplizarda

možda poglavlje iza poglavlja o konvolucijskom sloju?

The [convolutional layer](https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/) in convolutional neural networks systematically applies filters to an input and creates output feature maps.

Although the convolutional layer is very simple, it is capable of achieving sophisticated and impressive results. Nevertheless, it can be challenging to develop an intuition for how the shape of the filters impacts the shape of the output feature map and how related configuration hyperparameters such as padding and stride should be configured.

In this tutorial, you will discover an intuition for filter size, the need for padding, and stride in convolutional neural networks.

After completing this tutorial, you will know:

* How filter size or kernel size impacts the shape of the output feature map.
* How the filter size creates a border effect in the feature map and how it can be overcome with padding.
* How the stride of the filter on the input image can be used to downsample the size of the output feature map.

This tutorial is divided into five parts; they are:

1. Convolutional Layer
2. Problem of Border Effects
3. Effect of Filter Size (Kernel Size)
4. Fix the Border Effect Problem With Padding
5. Downsample Input With Stride

## CONVOLUTIONAL LAYER

In a convolutional neural network, a convolutional layer is responsible for the systematic application of one or more filters to an input.

The multiplication of the filter to the input image results in a single output. The input is typically three-dimensional images (e.g. rows, columns and channels), and in turn, the filters are also three-dimensional with the same number of channels and fewer rows and columns than the input image. As such, the filter is repeatedly applied to each part of the input image, resulting in a two-dimensional output map of activations, called a feature map.

Keras provides an implementation of the convolutional layer called a Conv2D.

It requires that you specify the expected shape of the input images in terms of rows (height), columns (width), and channels (depth) or *[rows, columns, channels]*.

The filter contains the weights that must be learned during the training of the layer. The filter weights represent the structure or feature that the filter will detect and the strength of the activation indicates the degree to which the feature was detected.

The layer requires that both the number of filters be specified and that the shape of the filters be specified.

## PROBLEM OF BORDER EFFECTS

In the previous section, we defined a single filter with the size of three pixels high and three pixels wide (rows, columns).

We saw that the application of the 3×3 filter, referred to as the kernel size in Keras, to the 8×8 input image resulted in a feature map with the size of 6×6.

That is, the input image with 64 pixels was reduced to a feature map with 36 pixels. Where did the other 28 pixels go?

The filter is applied systematically to the input image. It starts at the top left corner of the image and is moved from left to right one pixel column at a time until the edge of the filter reaches the edge of the image.

For a 3×3 pixel filter applied to a 8×8 input image, we can see that it can only be applied six times, resulting in the width of six in the output feature map.

(tu primjer od deeplizarda)

The reduction in the size of the input to the feature map is referred to as border effects. It is caused by the interaction of the filter with the border of the image.

This is often not a problem for large images and small filters but can be a problem with small images. It can also become a problem once a number of convolutional layers are stacked.

This can become a problem as we develop very deep convolutional neural network models with tens or hundreds of layers. We will simply run out of data in our feature maps upon which to operate.

## EFFECT OF FILTER SIZE (KERNEL SIZE)

Different sized filters will detect different sized features in the input image and, in turn, will result in differently sized feature maps.

It is common to use 3×3 sized filters, and perhaps 5×5 or even 7×7 sized filters, for larger input images. But, the 5×5 filter can only be applied to the 8×8 input image 4 times, resulting in a 4×4 feature map output.

## FIX THE BORDER EFFECT PROBLEM WITH PADDING

By default, a filter starts at the left of the image with the left-hand side of the filter sitting on the far left pixels of the image. The filter is then stepped across the image one column at a time until the right-hand side of the filter is sitting on the far right pixels of the image.

An alternative approach to applying a filter to an image is to ensure that each pixel in the image is given an opportunity to be at the center of the filter.

By default, this is not the case, as the pixels on the edge of the input are only ever exposed to the edge of the filter. By starting the filter outside the frame of the image, it gives the pixels on the border of the image more of an opportunity for interacting with the filter, more of an opportunity for features to be detected by the filter, and in turn, an output feature map that has the same shape as the input image.

For example, in the case of applying a 3×3 filter to the 8×8 input image, we can add a border of one pixel around the outside of the image. This has the effect of artificially creating a 10×10 input image. When the 3×3 filter is applied, it results in an 8×8 feature map. The added pixel values could have the value zero value that has no effect with the dot product operation when the filter is applied.

The addition of pixels to the edge of the image is called padding.

In Keras, this is specified via the “*padding*” argument on the Conv2D layer, which has the default value of ‘*valid*‘ (no padding). This means that the filter is applied only to valid ways to the input.

The ‘*padding*‘ value of ‘*same*‘ calculates and adds the padding required to the input image (or feature map) to ensure that the output has the same shape as the input.

## DOWNSAMPLE WITH STRIDE

The filter is moved across the image left to right, top to bottom, with a one-pixel column change on the horizontal movements, then a one-pixel row change on the vertical movements.

The amount of movement between applications of the filter to the input image is referred to as the stride, and it is almost always symmetrical in height and width dimensions.

The default stride or strides in two dimensions is (1,1) for the height and the width movement, performed when needed. And this default works well in most cases.

The stride can be changed, which has an effect both on how the filter is applied to the image and, in turn, the size of the resulting feature map.

For example, the stride can be changed to (2,2). This has the effect of moving the filter two pixels right for each horizontal movement of the filter and two pixels down for each vertical movement of the filter when creating the feature map.

This has the effect of applying the filter in such a way that the normal feature map output (6×6) is down-sampled so that the size of each dimension is reduced by half (3×3), resulting in 1/4 the number of pixels (36 pixels down to 9).

*The stride can be specified in Keras on the Conv2D layer via the ‘stride‘ argument and specified as a tuple with height and width.*

Primjeri ili odavde ili deeplizard

Downsampling may be desirable in some cases where deeper knowledge of the filters used in the model or of the model architecture allows for some compression in the resulting feature maps.

## Summary

In this tutorial, you discovered an intuition for filter size, the need for padding, and stride in convolutional neural networks.

Specifically, you learned:

* How filter size or kernel size impacts the shape of the output feature map.
* How the filter size creates a border effect in the feature map and how it can be overcome with padding.
* How the stride of the filter on the input image can be used to downsample the size of the output feature map.