# Pooling Layer Basics

compiled by D.Gueorguiev, 6/9/2024

## Preliminaries

### Convolution

Continuous space convolution (abbrev. continuous convolution)

(1)

Shorthand notation for the convolution operator is :

(2)

Think of the function as an *input* and the second function as a *kernel*. The output is sometimes referred to as the *feature map*.

Discrete space convolution (abbrev. discrete convolution)

(3)

With discrete convolution the input is multi-dimensional array of data, and the kernel is a multi-dimensional array of parameters related to the specific learning algorithm being used. These multi-dimensional arrays will be referred to as *tensors*.

Often, we will use discrete convolutions over more than one axis. For example for two-dimensional image as an output we would use two-dimensional kernel in general:

(4)

(4) can be rewritten as:

(5)

Thus we conclude that discrete convolution is commutative i.e.

(6)

Flipping operator :

(7)

Using the notation introduced with (7) we rewrite (4) and (5) as

(8)

(9)

Cross-correlation

The same as discrete convolution but the kernel is not *flipped*:

(10)

We will denote the *cross-correlation* operator with .

Discrete convolution can be viewed as a multiplication by a matrix.

## Pooling Layers

### Example: Vertical Line Detector

Below it is presented a small example which takes as an input an image of size 8 pixels by 8 pixels with one channel and detects if it finds vertical line.

# example of vertical line detection with a convolutional layer

from numpy import asarray

from keras.models import Sequential

from keras.layers import Conv2D

# define input data

data = [[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]]

data = asarray(data)

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

# create model

model = Sequential()

model.add(Conv2D(1, (3,3), activation='relu', input\_shape=(8, 8, 1)))

# summarize model

model.summary()

# define a vertical line detector

detector = [[[[0]],[[1]],[[0]]],

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

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

weights = [asarray(detector), asarray([0.0])]

# store the weights in the model

model.set\_weights(weights)

# apply filter to input data

yhat = model.predict(data)

# enumerate rows

for r in range(yhat.shape[1]):

# print each column in the row

print([yhat[0,r,c,0] for c in range(yhat.shape[2])])

## Appendix

### Toeplitz matrix (aka diagonal-constant matrix)

A matrix in which each descending diagonal from left to right is constant. For example:

A matrix constructed such that is another example of Toeplitz matrix.

## References

[Pooling Layers for Convolutional Neural Networks, Jason Brownlee, July 5, 2019, online tutorial](https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/)

[Deep Learning for Computer Vision: Image Classification, Object Detection, Face Recognition, Jason Brownlee, 2019](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/books/Deep_Learning_for_Computer_Vision-Image_Classification_Object_Detection_and_Face_Recognition_in_Python_by_Jason_Brownlee.pdf)

[Deep learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016](file:///Users/dimitargueorguiev/git/ml/deep_learning_for_image_processing/literature/books/deeplearning_latest_edition.pdf)