

CPSC 532P / LING 530A: Deep Learning for Natural Language Processing (DL-NLP)

Muhammad Abdul-Mageed

`muhammad.mageed@ubc.ca`

Natural Language Processing Lab

The University of British Columbia

Table of Contents

- 1 CNN Applications
- 2 What Are ConvNets?
- 3 2D Convolution
- 4 Pooling
- 5 Motivations for Convolution

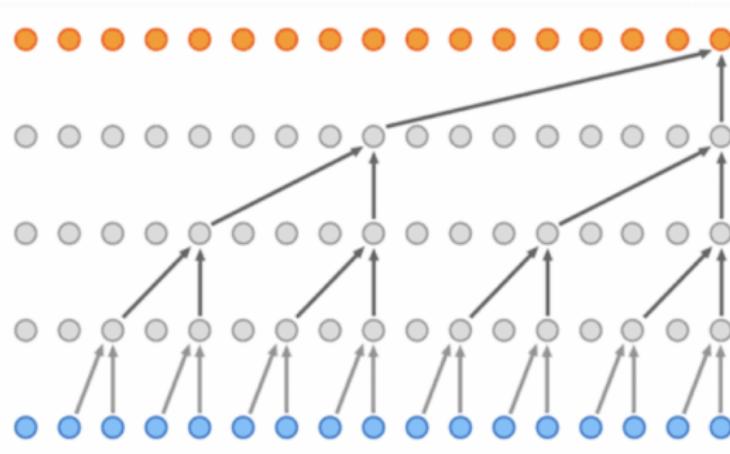
ConvNets All Over The Place



Figure: CNN applications in self-driving cars, art, and celebrity face-generation.

ConvNets In Speech Synthesis

WaveNet: A Generative Model for Raw Audio



See: WaveNet.

ConvNets In Medical Applications

Improved Grading of Prostate Cancer Using Deep Learning

Friday, November 16, 2018

Posted by Martin Stumpe, Technical Lead and Craig Mermel, Product Manager, Healthcare, Google AI

Approximately **1 in 9 men** in the United States will develop prostate cancer in their lifetime, making it the **most common cancer in males**. Despite being common, prostate cancers are frequently non-aggressive, making it challenging to determine if the cancer poses a significant enough risk to the patient to warrant treatment such as surgical removal of the prostate (**prostatectomy**) or radiation therapy. A key factor that helps in the **"risk stratification"** of prostate cancer patients is the **Gleason grade**, which classifies the cancer cells based on how closely they resemble normal prostate glands when viewed on a slide under a microscope.

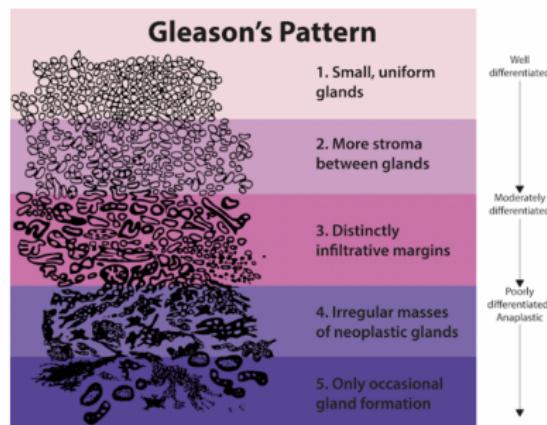


Jeff Dean ✨
@JeffDean

Following

New work on pathology from
@GoogleAI:

"... the mean accuracy among 29 general pathologists was 0.61. The DLS achieved a[n]... accuracy of 0.70 (p=0.002) and trended towards better patient risk stratification..."



See: Google AI Blog.

What Are ConvNets?

- ConvNets (LeCun, 1989) classically specialized for processing **data that has a known grid-like typology**.
- Example 1: Time-series data conceptualized as a 1-D grid taking examples at regular intervals.
- Example 2: Image data, which can be thought of as a 2-D grid of pixels.
- Employ a mathematical operation called **convolution**
- Convolution is a specialized kind of ***linear*** operation
- ConvNets are simply **neural nets that use convolution in place of general matrix multiplication in at least one of their layers**

Convolution Explained

Tracking a Spaceship! (Example from Goodfellow et al. (2016))

- We are **tracking the location of a spaceship** with a laser sensor outputting the position x of the ship at time t .
- Both x and t are real-valued, meaning we **can get a different reading from the sensor at any given moment** in time.



Figure: [From www.starwars.com/].

Convolution: The Spaceship Example

Measuring Location of a Moving Spaceship

- **Problem:** Our laser sensor is noisy.
- **Solution:** We need to average several measurements.
- **Observation:** More recent measurements are more relevant.
- **Modified solution:** Get weighted average that assigns more weight to more recent observations.
- Use a **weighting function** $w(a)$, where a is the age of measurement.
- Applying $w(a)$, we obtain a new function s that provides a **smoothed estimate of the position of the spaceship**.
- Thus, in its most general form, convolution is an **operation on two functions of a real-valued argument**
- In this example, we have **(1) a weighting function w and (2) a measurement function (with averaging mechanism) s**

Convolution as Weighted Averaging Over Moments

Convolution Over Moments

- The **convolution** operation is usually denoted with an asterisk:
 $s(t) = (x * w)(t)$.
- Weighting function **w** is a **probability density function**, with **a** being the age of measurement:

1: Convolution | Weighted avg. over moment a

$$s(t) = \int x(a)w(t - a)da.$$

Discrete Convolution

Discrete Convolution

- With discretized time (with t expressed as an integer), we have discrete convolution:

2: Discrete Convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a).$$

Convolution Terminology

- The function x : the **input**
- The function w : the **kernel**
- The **output**: a **feature map**

2D Convolution

5 x 5 matrix (e.g., input image)

1	1	0	0	0
0	0	1	1	1
0	0	0	0	0
1	1	0	0	1
1	1	0	1	1

3 x 3 kernel (aka filter)

1	0	0
0	1	1
1	0	0

Desired feature map

Figure: **Input:** 5 x 5 matrix/2D Tensor. We will convolve with a 3 x 3 **kernel**, with a **stride** of 1. Our goal is to acquire an **feature map** (filling in the 3 x 3 matrix on right-hand side).

Convolving 1

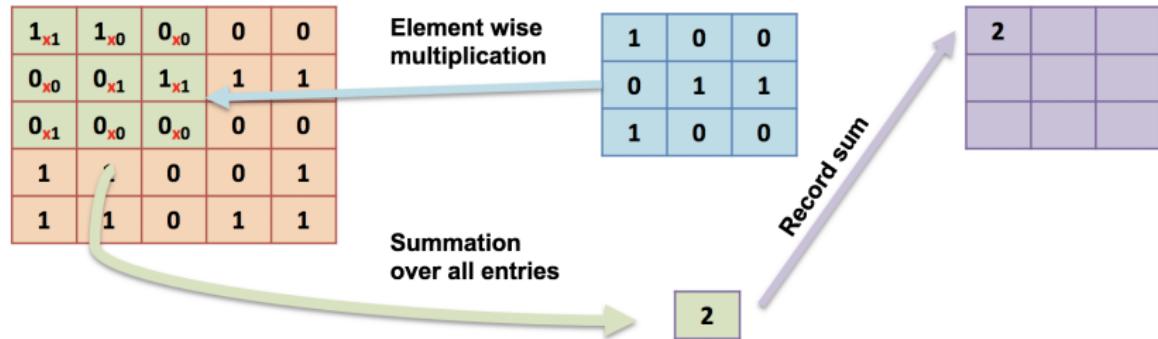


Figure: Filling in the first cell.

Convolving 2

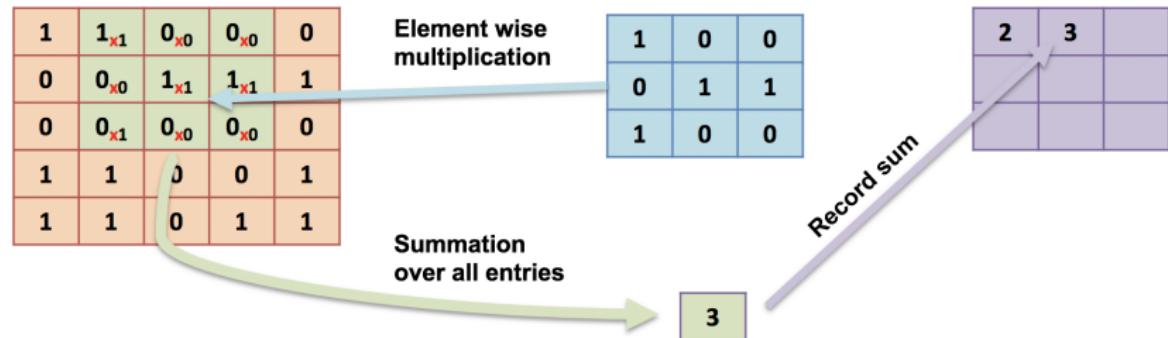


Figure: Filling in the second cell.

Convolving 3

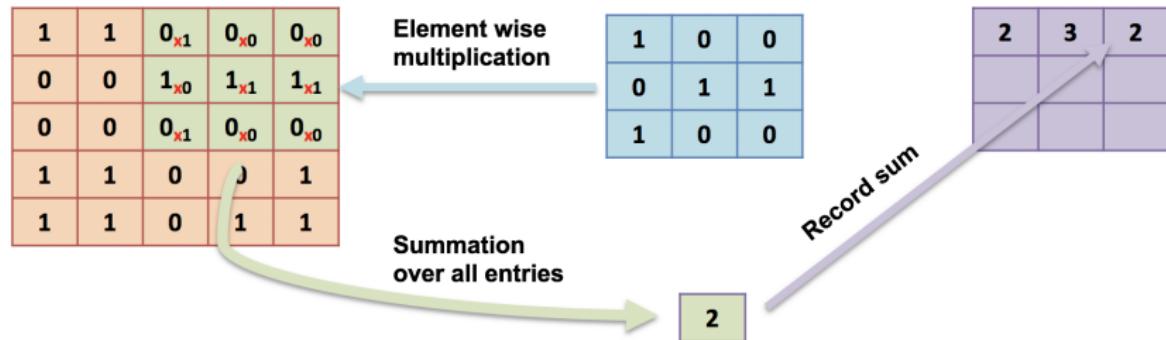


Figure: Filling in the third cell.

Convolving 4

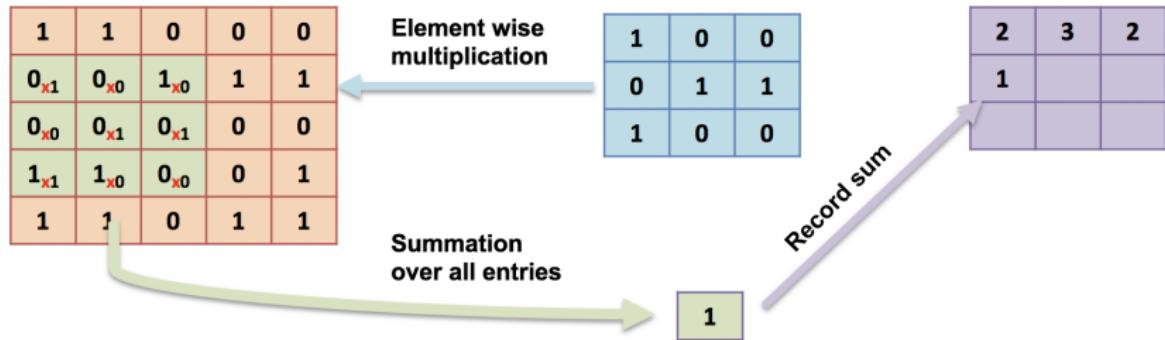


Figure: Filling in the fourth cell.

Convolving 5-9

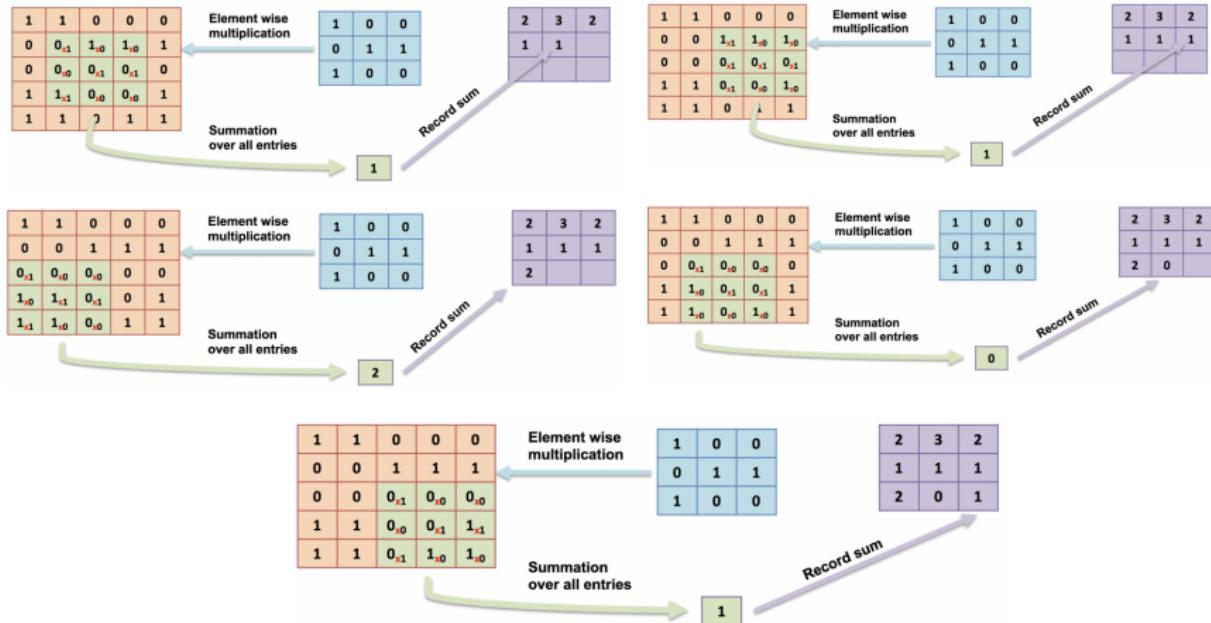


Figure: Filling in the fifth-to-ninth cells.

Convolving With 10 Kernels

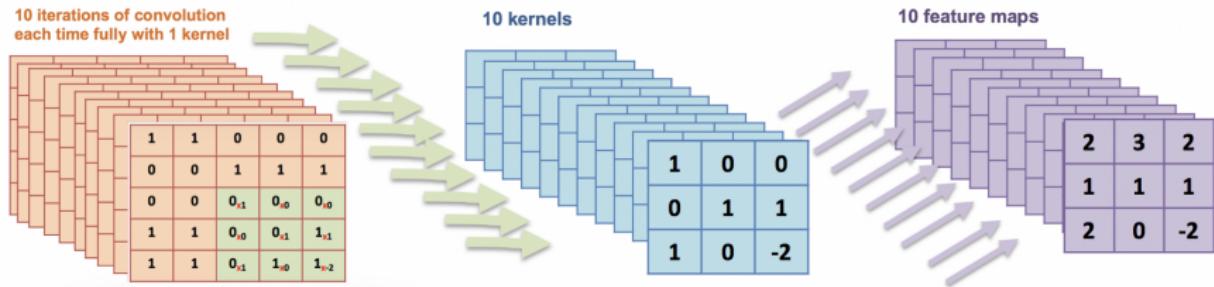


Figure: Convolving with 10 kernels, each with different values, we acquire 10 different feature maps.

Non-Linearity

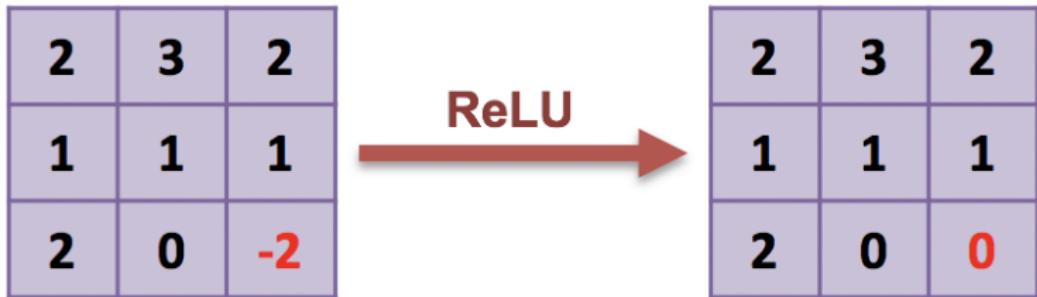


Figure: We actually apply a non-linear function like ReLU on the output of convolution operations such that we end up with meaningful feature maps.

Convolution With Non-Linearity

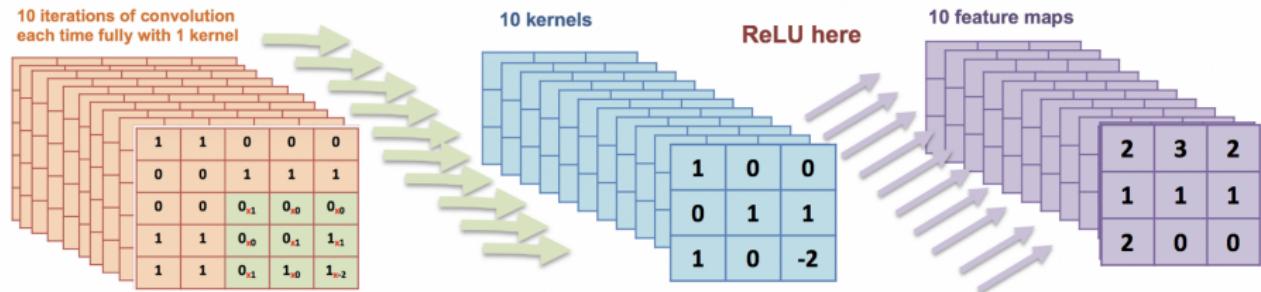


Figure: We actually apply a non-linear function like ReLU on the output of convolution operations such that we end up with meaningful feature maps.

A larger feature map

0.36	0.13	0.72	0.59	0.31	0.23	0.79	0.47
0.35	0.28	0.65	0.52	0.73	0.24	0.76	0.29
0.63	1	0.23	0.06	0.47	0.06	0.85	0.29
0.79	0.47	0.97	0.15	0.15	0.66	0.28	0.61
0.43	0.08	0.03	0.07	0.54	0.33	0.43	0.11
0.66	0.35	0.26	0.33	0.99	0.18	0.9	0.9
0.29	0.79	0.43	0.24	0.41	0.87	0.3	0.12
0.25	0.38	0.44	0.78	0.72	0.77	0.02	0.12



1	0	-1
0	1	1
1	0	-1



0.65	0.97	-0.55	-0.3	0.36	0.93	0.46	0.95
0.11	0.59	0.98	0.35	0.86	0.96	0.13	0.09
0.07	0.2	0.58	0.06	0.64	0.99	0.4	0.33
0.78	0.67	-0.88	-0.44	0.55	0.28	0.63	0.14
0.83	0.45	0.7	0.4	0.3	0.12	0.04	0.98
0.61	0.07	0.2	0.55	0.95	0.94	0.65	0.62
0.62	0.55	0.91	0.85	0.17	0.66	0.92	0.15
0.42	0.63	0.95	0.13	0.15	0.51	0.22	0.38

Figure: A larger feature map, still with negative values.

ReLU Applied to Feature Map

0.36	0.13	0.72	0.59	0.31	0.23	0.79	0.47
0.35	0.28	0.65	0.52	0.73	0.24	0.76	0.29
0.63	1	0.23	0.06	0.47	0.06	0.85	0.29
0.79	0.47	0.97	0.15	0.15	0.66	0.28	0.61
0.43	0.08	0.03	0.07	0.54	0.33	0.43	0.11
0.66	0.35	0.26	0.33	0.99	0.18	0.9	0.9
0.29	0.79	0.43	0.24	0.41	0.87	0.3	0.12
0.25	0.38	0.44	0.78	0.72	0.77	0.02	0.12



1	0	-1
0	1	1
1	0	-1



0.65	0.97	0.6	0.48	0.36	0.93	0.46	0.95
0.11	0.59	0.98	0.35	0.86	0.96	0.13	0.09
0.07	0.2	0.58	0.06	0.64	0.99	0.4	0.33
0.78	0.67	0.91	0.59	0.55	0.28	0.63	0.14
0.83	0.45	0.7	0.4	0.3	0.12	0.04	0.98
0.61	0.07	0.2	0.55	0.95	0.94	0.65	0.62
0.62	0.55	0.91	0.85	0.17	0.66	0.92	0.15
0.42	0.63	0.95	0.13	0.15	0.51	0.22	0.38

Figure: Feature map from previous slide, with ReLU applied.

Several Feature Maps

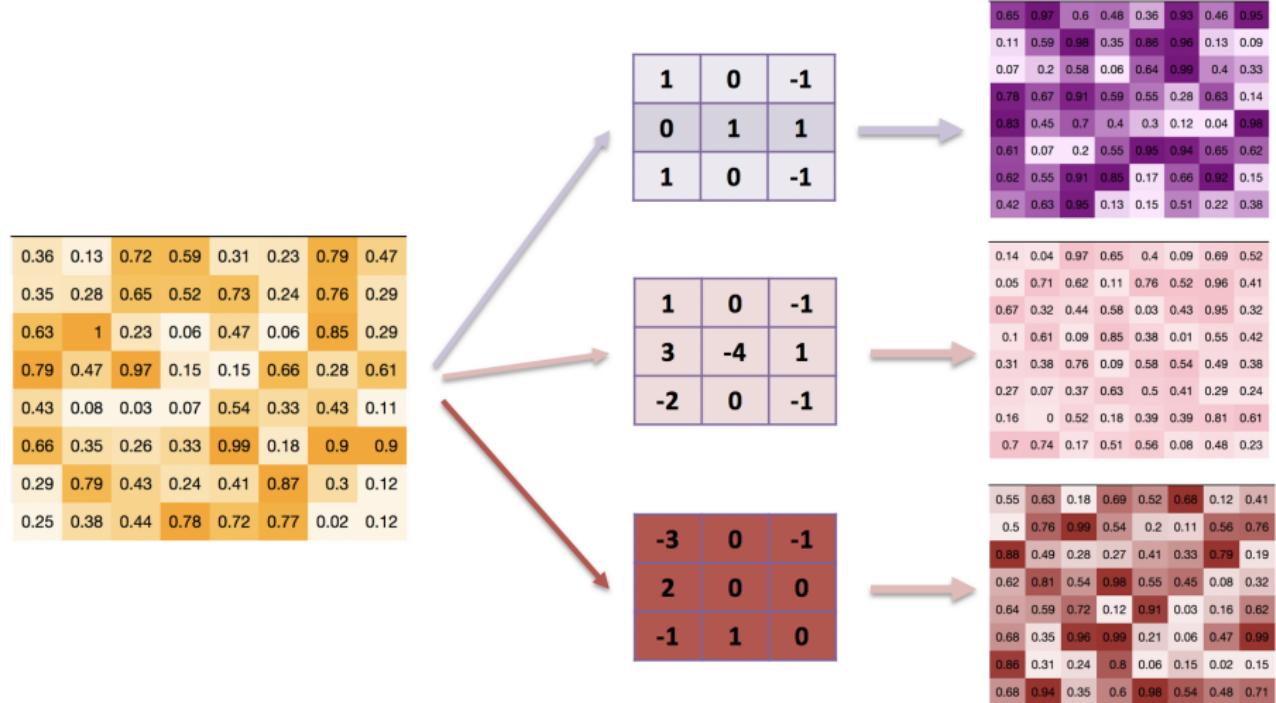


Figure: We acquire several feature maps (number is a hyperparameter) from input.

Feature Maps Stacked

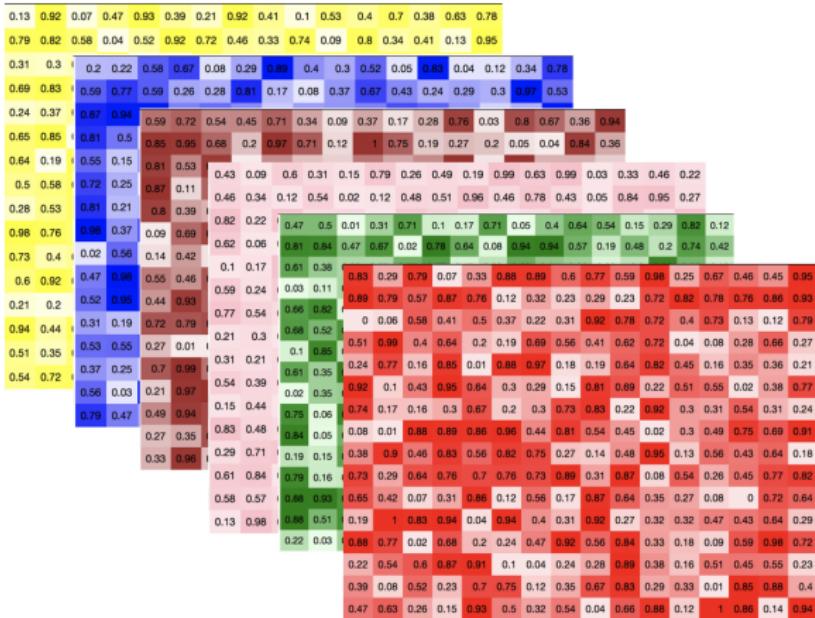


Figure: Several feature maps stacked

Max Pooling

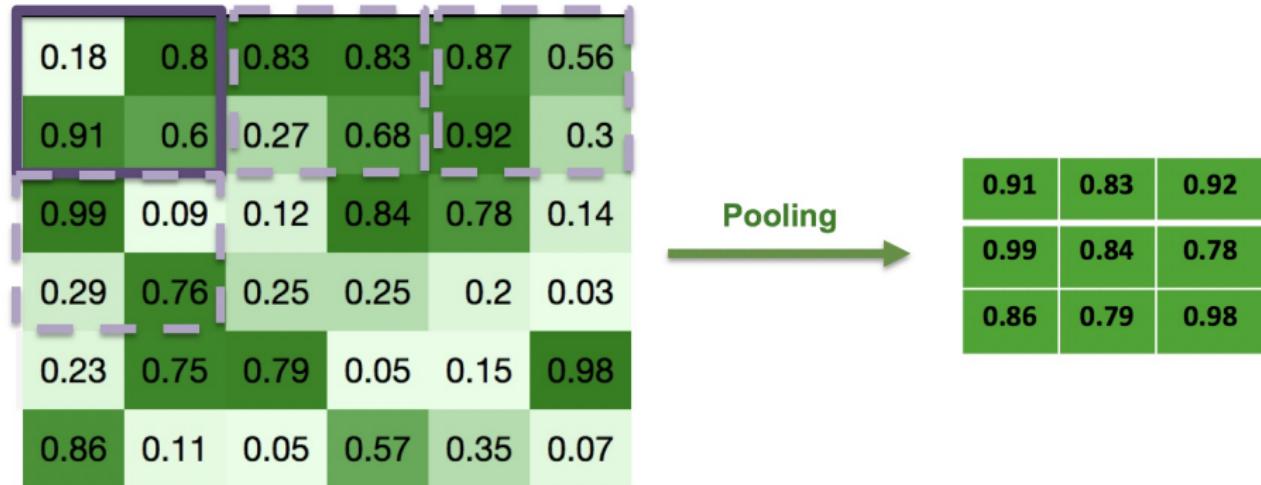


Figure: Max pooling (with a stride of 2) on one feature map

Bigger Picture

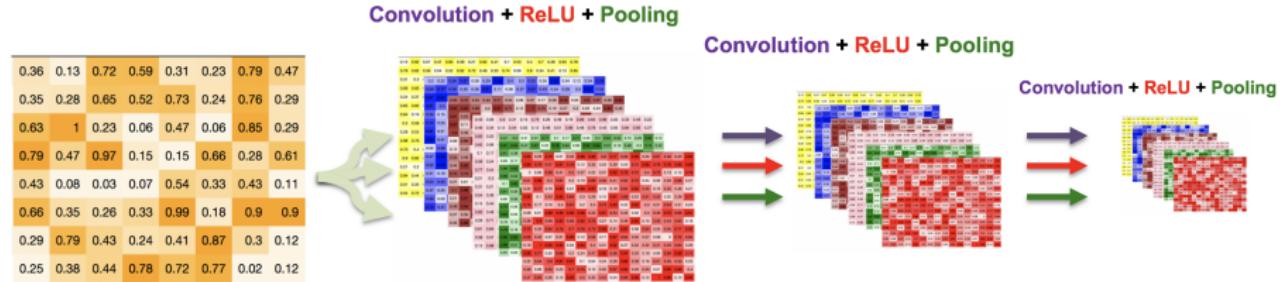


Figure: Convolution and pooling layers, performed on several feature maps

Fully Connected Layer



Figure: We collapse last layer of feature maps to one fully connected layer.

Network With Fully Connected Layer

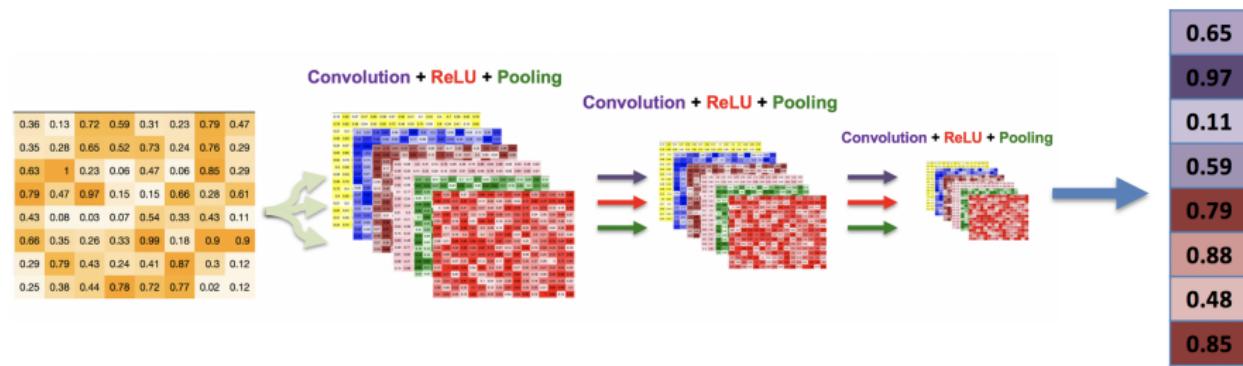


Figure: The network with a fully connected layer.

Motivations for Convolution

- In addition to it providing a means to **work with inputs of variable size**, convolution offers a number of advantages:

Advantages of Convolution

- **Sparse interactions**
- **Parameter sharing**
- **Equivariant representations**

Sparse Interactions

Sparse Interactions

- CNN uses a **kernel smaller than its input units**, giving **outputs that are not the result of an interaction with each input unit**.
- **Huge computational gains** (we need to store fewer parameters)

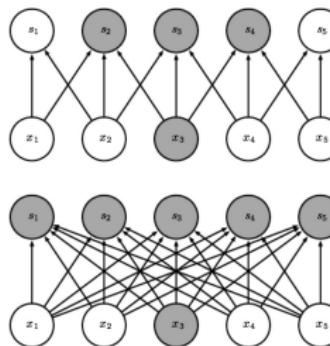


Figure: (Top): When s is formed by **convolution**, with a kernel of width 3, only 3 output units are affected by x . (Bottom): When s is formed by matrix multiplication, **connectivity is no longer sparse**. [Goodfellow et al. (2016, p. 325)].

Parameter Sharing

Parameter Sharing

- In a traditional neural net, each element of the weight matrix is multiplied by one element of the input and then **never revisited** (never used again).
- ConvNets have **shared** (or **tied weights**), where **each member of the kernel is used at every position of the input**.

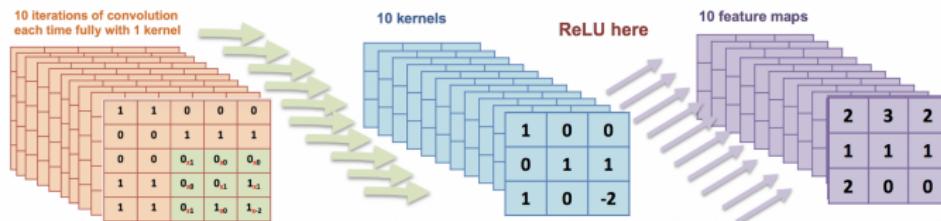


Figure: Each member of a given kernel is used at every position of the input

Equivariance

- The form of parameter sharing in convolution causes the layer to have equivariance.
- Equivariant representations (or equivariance to translation) in a given function means that if the input changes, the output changes the same way.
- If a feature moves later in input, its exact same representation will still appear in the output, just later
- Useful for image, when we detect edges, since the same edges will appear in different parts.
- Can be useful in language, e.g., when we move character sequences.

Example CNN For Sentence Classification

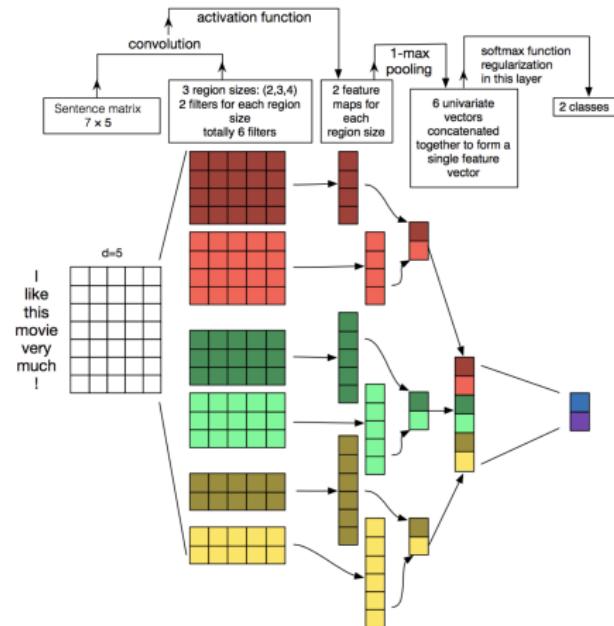


Figure: [From Zhang, Y., Wallace, B. (2015)]