Introduction to Convolutional Neural Networks

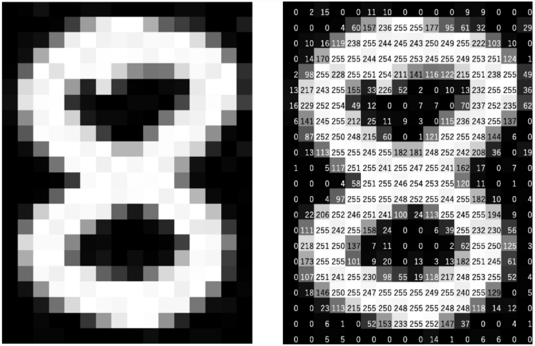
Ryan Miller

Introduction

- The neural network architectures from our previous lecture/lab are limited by the fact that they do not account for spatial structure within the input data
- Convolutional neural networks use the mathematical operation of *convolution* to identify and extract spatially dependent hidden features
    - This is particularly effective for image data, but can be used for any data where the relative positions of input features is meaningful

MNIST Example

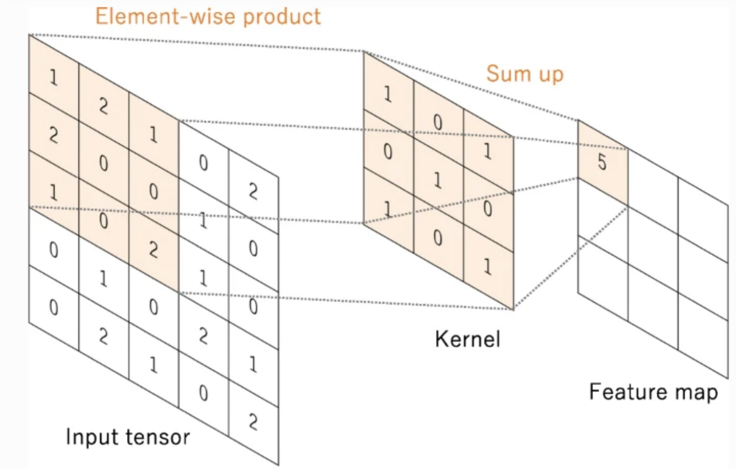
To understand convolution, let's start with an example image from the MNIST data:



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

How would a "vanilla" neural network learn that this example is an eight?

Convolution takes a *matrix of weights* known as a **kernel** (or filter) and slides it across an input matrix to generate a **feature map**:

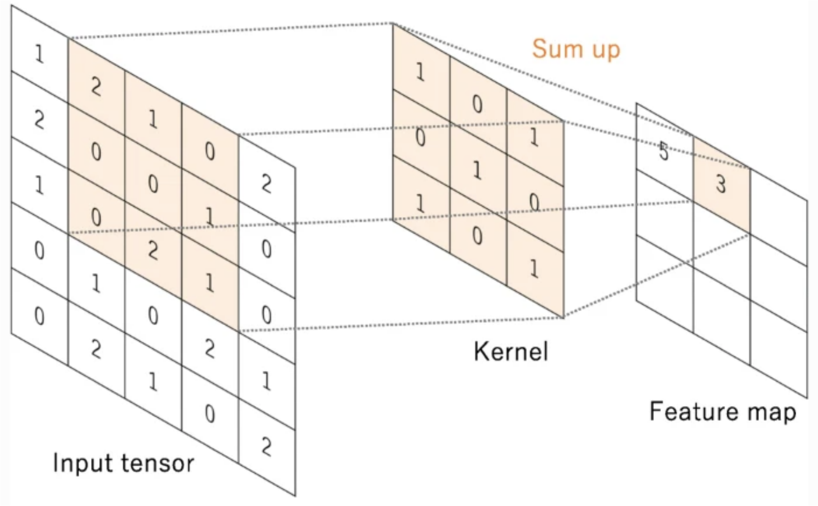


Stride

- **Stride** describes how a convolution kernel moves across the input to generate the feature map
  - A stride of 1 will move the kernel 1 element at a time (in the direction of rows or columns)
  - A stride of 2 will move the kernel 2 elements at a time
- Larger stride will decrease the dimensions of the feature map produced by convolution, but can also result in a loss of information (if the location where a pattern is most prominent gets passed over)

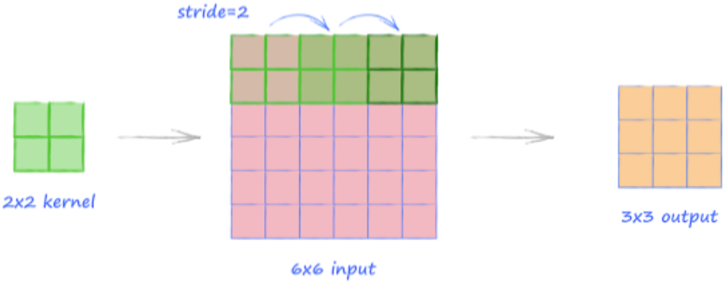
Stride

The diagram in our earlier example uses a stride of 1. Here is the next element in the feature map:



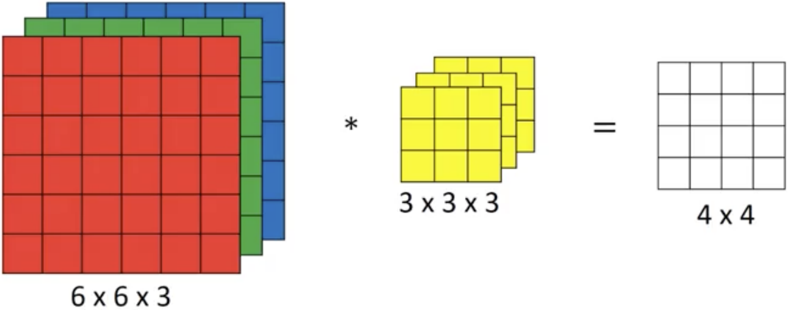
Stride

The diagram below demonstrates a stride of 2. We can see that this produces a much smaller feature map:



Multiple Input Channels

For inputs with multiple channels (such as images with RGB channels), convolution can use a separate kernel for each channel:



Remarks on Convolution

- The main benefit of convolution is that kernels are shared across multiple locations of the input
  - Thus, hidden features can be learned from different locations
- In contrast, the hidden features learned in the “vanilla” neural network architectures we’ve previously discussed are position sensitive
  - For example, a neuron might learn a horizontal edge, but it could only do so in a specific combination pixel positions
  - If the data were rigorously preprocessed this might be okay, but in general it’s a major limitation

Example (horizontal edges)

1	2	1
0	0	0
-1	-2	-1

0	0	0	0	0	0	2	1	0	0	0	0	0	0	1	9	18	3	0	1	8	0	0	0	0	0
0	0	0	0	0	0	0	0	11	0	0	28	20	0	0	0	0	4	23	8	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	13	7	0	0	13	8	1	0	0	0	0	0	7	14	7	0	0
0	0	0	0	0	0	9	18	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	30	##	##	##	##	##	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	38	##	##	##	##	##	##	##	##	##	0	0	0	0	0	0	0
4	7	6	0	0	0	0	0	0	##	##	##	##	##	##	##	##	##	##	18	25	0	0	0	0	0
0	5	6	3	8	20	97	##	84	0	0	##	##	##	41	0	0	0	0	##	##	##	56	10	0	0
0	0	0	2	16	39	##	##	##	0	0	0	0	0	0	0	0	0	0	##	##	##	40	0	0	0
5	0	0	0	0	9	##	##	##	##	0	0	0	0	0	0	0	0	##	##	##	99	0	0	0	0
4	5	0	0	0	0	83	##	##	##	99	0	0	0	0	0	0	##	##	##	##	10	20	10	0	0
0	4	10	10	5	1	13	##	##	##	##	##	0	0	0	0	##	##	##	##	53	37	19	0	0	0
0	1	5	10	11	8	4	2	22	37	15	0	0	7	9	4	0	0	0	0	0	5	8	6	0	0
5	1	0	0	0	0	0	0	0	0	0	0	##	##	##	##	27	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	##	##	##	##	##	##	##	0	0	0	0	4	0	0	0
0	0	0	0	0	0	0	0	0	80	##	##	##	##	##	##	##	##	##	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	##	##	##	##	0	0	19	##	##	32	0	0	0	0	0	0	0
0	0	0	0	0	0	##	##	##	0	0	0	0	0	0	0	0	0	0	77	##	80	17	9	0	0
0	0	0	0	0	0	##	##	##	0	0	0	0	0	0	0	0	0	62	##	##	42	0	0	0	0
0	0	0	0	0	0	##	##	##	##	##	0	0	0	0	0	0	##	##	##	##	48	4	0	0	0
0	0	0	0	0	0	18	##	##	##	##	##	##	##	31	##	##	##	##	##	##	14	24	10	0	0
0	0	0	0	0	0	0	18	##	##	##	##	##	##	##	##	##	##	##	##	32	11	0	0	0	0
0	0	0	0	0	0	0	2	11	6	45	##	##	##	##	##	##	##	33	2	9	0	0	0	0	0
0	0	0	0	0	0	0	0	3	11	0	0	0	0	1	0	0	0	13	5	0	0	0	1	0	0

0	0	0	2	11	28	20	9	18	23	8	0	0
0	0	0	18	13	7	13	8	0	0	7	14	0
0	0	0	0	0	0	0	0	0	0	0	12	0
0	0	0	0	38	##	##	##	##	##	0	0	0
7	6	20	##	##	##	##	##	##	##	##	56	0
5	2	39	##	##	0	0	0	0	##	##	40	0
5	10	5	##	##	##	0	0	##	##	##	20	0
5	10	11	4	37	15	##	##	27	0	5	8	0
0	0	0	0	80	##	##	##	##	##	0	4	0
0	0	0	##	##	##	##	19	##	77	##	17	0
0	0	0	##	##	##	0	0	##	##	##	4	0
0	0	0	##	##	##	##	##	##	##	##	24	0
0	0	0	2	11	##	##	##	##	33	9	1	0

Example (vertical edges)

1	0	-1
2	0	-2
1	0	-1

0	0	0	0	0	0	0	0	0	15	0	0	21	1	15	11	0	0	0	23	0	0	8	9	0	0
0	0	0	0	0	0	0	0	0	0	6	0	0	0	13	13	9	0	0	20	3	0	4	20	0	0
0	0	0	0	0	0	0	0	5	0	4	12	0	0	5	6	13	7	1	11	8	0	0	19	0	0
0	0	0	0	0	0	0	0	7	0	0	0	0	0	77	##	##	56	75	48	0	0	29	6	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	##	##	##	##	##	##	45	10	58	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	##	##	##	##	##	##	##	30	8	0	0
0	0	4	2	0	0	0	0	0	0	0	0	28	##	##	9	0	0	##	##	##	##	51	19	0	0
0	0	11	5	0	0	0	0	0	3	##	0	50	##	##	28	0	0	0	98	##	##	##	24	0	0
0	0	12	1	0	0	0	0	0	##	##	0	14	##	##	10	0	0	0	0	##	##	##	28	0	0
0	3	8	0	0	0	0	0	0	##	##	##	21	##	47	0	0	0	0	87	##	##	##	20	0	0
0	5	6	0	0	0	0	0	0	##	##	##	##	68	0	0	0	0	0	##	##	##	62	21	0	0
0	0	3	4	1	0	0	0	0	0	##	##	##	##	0	0	0	0	##	##	##	##	19	32	0	0
0	0	0	3	7	4	0	0	0	0	0	##	##	##	0	0	0	##	##	##	##	0	38	22	0	0
2	0	0	0	5	5	0	0	0	0	0	0	68	63	0	0	##	##	##	##	17	3	19	12	0	0
6	1	0	0	1	2	2	0	0	0	0	0	17	0	0	12	##	##	##	##	6	3	4	6	0	0
3	1	0	0	0	0	0	0	0	0	0	0	##	##	0	0	1	##	##	##	##	2	8	8	0	0
0	0	0	0	0	0	0	0	0	0	0	##	##	##	0	0	0	0	##	##	##	58	4	16	0	0
0	0	0	0	0	0	0	0	0	74	##	##	##	##	0	0	0	0	0	##	##	##	3	17	0	0
0	0	0	0	0	0	0	0	0	##	##	##	##	53	0	0	0	0	0	##	##	##	6	19	0	0
0	0	0	0	0	0	0	0	0	##	##	##	##	##	0	0	0	0	0	##	##	##	7	11	0	0
0	0	0	0	0	0	0	0	0	37	##	##	##	##	0	0	0	0	34	##	##	##	13	11	0	0
0	0	0	0	0	0	0	0	0	0	0	##	##	74	0	0	0	94	##	##	##	44	14	20	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	##	##	##	##	##	##	18	6	10	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	##	##	##	##	##	18	19	2	1	0	0
0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	##	##	##	25	10	11	0	1	8	0	0
0	0	0	0	0	0	0	0	9	0	0	1	0	0	0	22	26	0	0	3	6	0	3	13	0	0

0	0	0	0	15	6	21	15	9	23	3	20	0
0	0	0	0	7	12	0	##	##	75	8	29	0
0	0	0	0	0	0	0	##	##	##	##	58	0
0	11	0	0	3	##	##	##	0	##	##	##	0
0	12	0	0	##	##	##	##	0	87	##	##	0
0	6	1	0	##	##	##	0	0	##	##	62	0
0	3	7	0	0	##	##	0	##	##	##	38	0
3	0	2	2	0	0	##	12	##	##	##	8	0
5	0	0	0	74	##	##	0	0	##	##	17	0
5	0	0	0	##	##	##	0	0	##	##	19	0
0	0	0	0	37	##	##	0	94	##	##	20	0
2	0	0	0	0	0	0	##	##	##	##	10	0
6	0	0	0	14	1	0	##	##	25	11	13	0

Padding

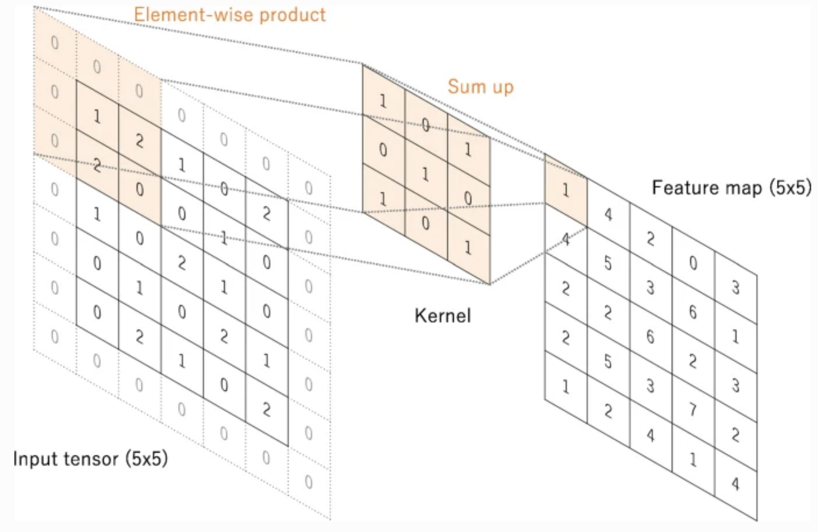
- Convolution does not allow the center of a kernel to pass over the edges of the input tensor
  - There are some benefits to this, as it reduces the dimension of the feature map relative to the input tensor
  - However, if the edges contain an important feature it can be problematic

Padding

- Convolution does not allow the center of a kernel to pass over the edges of the input tensor
  - There are some benefits to this, as it reduces the dimension of the feature map relative to the input tensor
  - However, if the edges contain an important feature it can be problematic
- **Padding** addresses this issue by adding extra rows and columns to create an artificial border around the input tensor
  - This can allow the center of the convolution kernel to pass over the edges of an image (or feature map)
  - *Zero padding*, which fills the extra rows with zeros, is the most common type of padding

Padding

The diagram below illustrates zero padding:

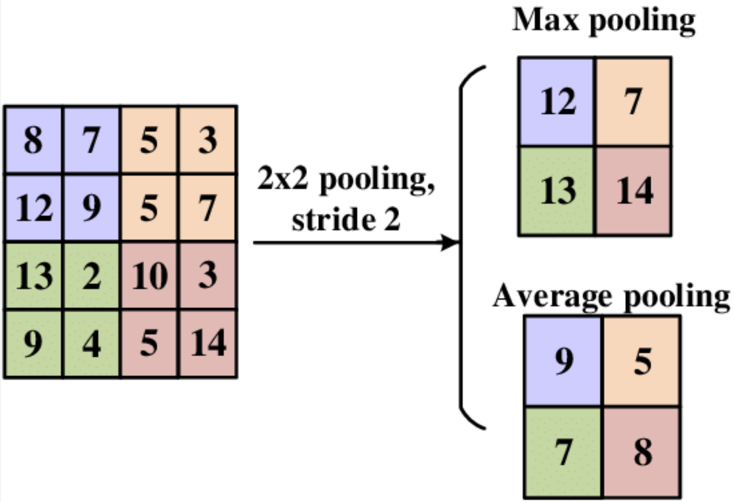


Pooling

- Convolutional layers can quickly increase the size and complexity of a network
- While increasing stride can help reduce the size of feature maps, an operation known as *pooling* tends to be more popular
  - *Max pooling*, which keeps the maximum value in each "patch" of an input feature map is most commonly used
  - *Average pooling*, which keeps the average value within a "patch" is sometimes used

Pooling

Pooling operations require a patch size and stride. Shown below is pooling using a 2x2 patch with a stride of 2:

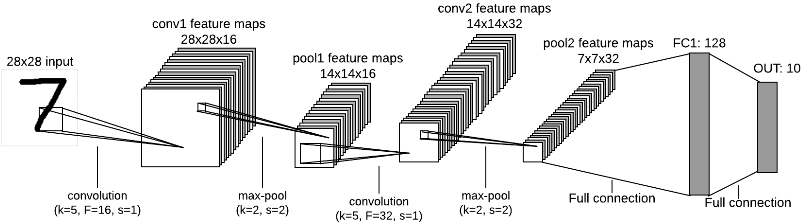


Max vs. Average Pooling

Max pooling is widely viewed as superior because it highlights and retains the most salient spatial features:

Architecture

The diagram below demonstrates what the architecture of a convolutional neural network might look like:



Remarks on Architecture

- Near the end of a convolutional neural network the feature maps are flattened
  - It's logical to do this when the feature map no longer contains meaningful spatial information
  - It's also necessary to produce a properly formatted output vector
- Pooling helps keep the number of parameters in the network under control
  - For example, AlexNet used roughly 60 million parameters despite including 3 different pooling layers

Image Credits

Links to images used in this presentation can be found below:

- "Convolutional neural networks: an overview and application in radiology"
- Max pooling vs Average Pooling
- CNN architecture example