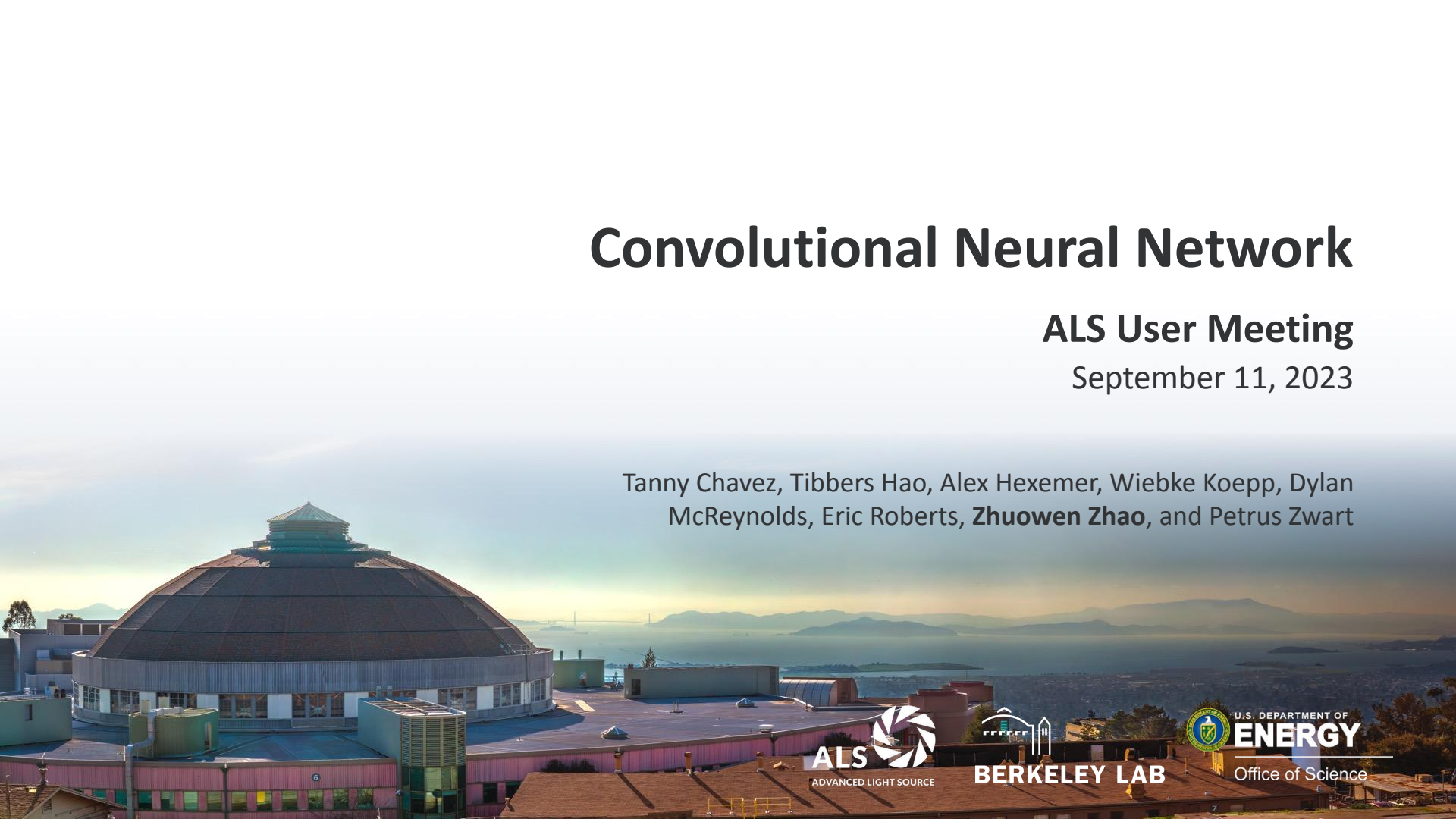
Convolutional Neural Network

ALS User Meeting

September 11, 2023

Tanny Chavez, Tibbers Hao, Alex Hexemer, Wiebke Koepp, Dylan McReynolds, Eric Roberts, **Zhuowen Zhao**, and Petrus Zwart



What are convolutional neural networks?

Convolutional neural networks are distinguished from other neural networks by their superior performance with image inputs. They have 3 main types of layers: the convolutional layer, pooling layer, and fully-connected (FC) layer.

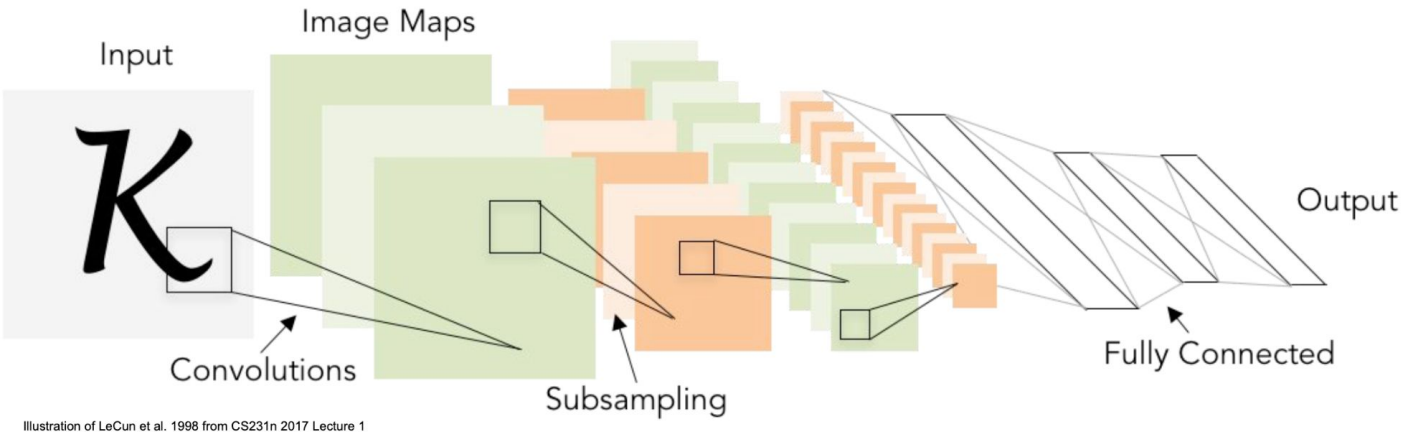
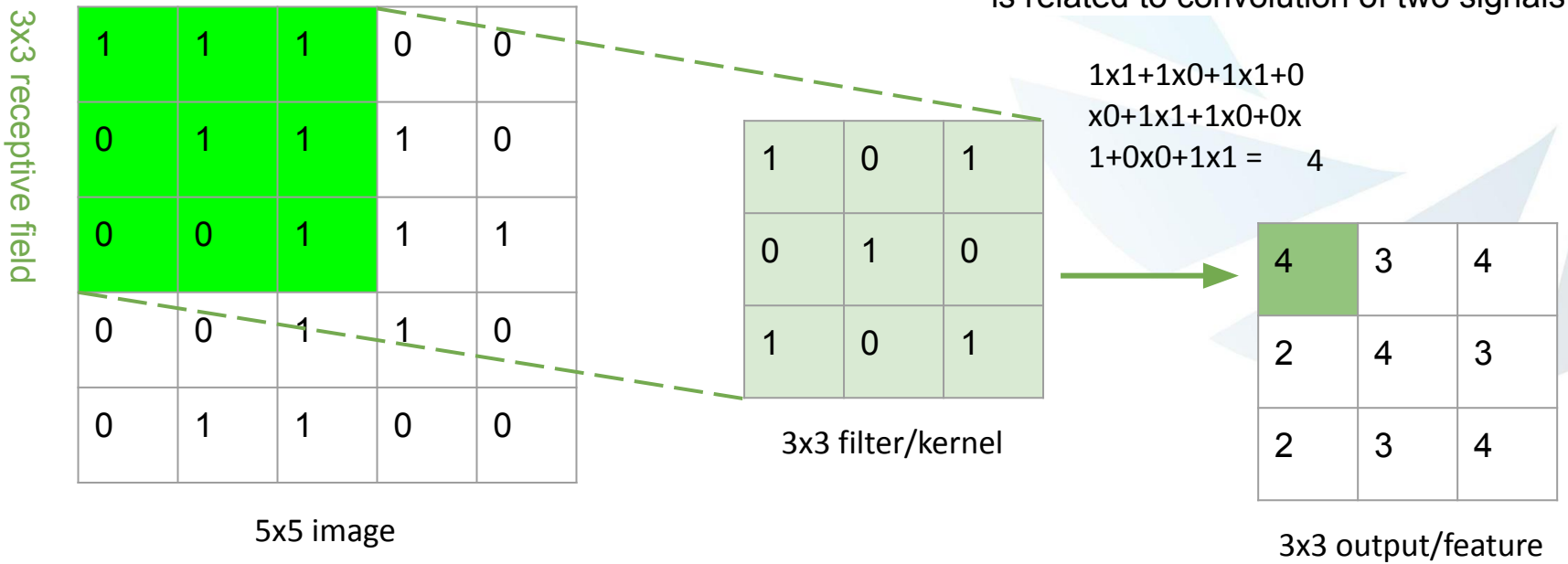


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Convolutional layer

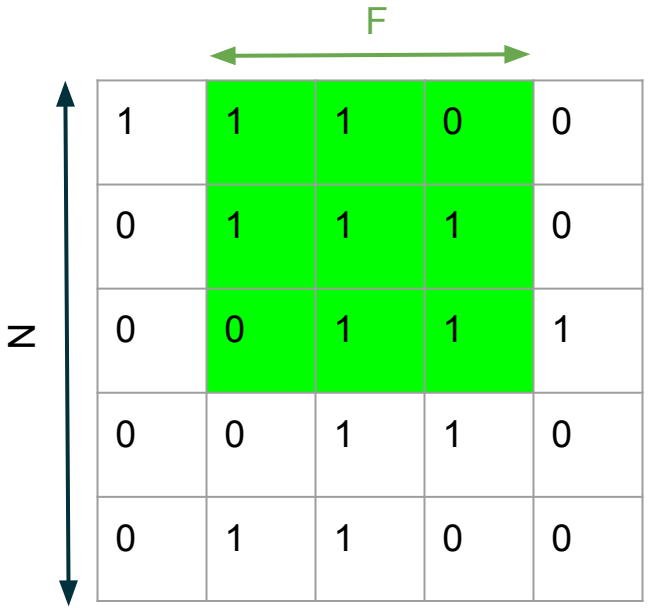
The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

We call the layer convolutional because it is related to convolution of two signals



Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.



5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

stride (step size) = 1

output size = (N-F)/stride + 1

4	3	4
2	4	3
2	3	4

3x3 output/feature

Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

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

5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

4	3	4
2	4	3
2	3	4

3x3 output/feature

Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

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

5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

4	3	4
2	4	3
2	3	4

3x3 output/feature

Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

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

5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

4	3	4
2	4	3
2	3	4

3x3 output/feature

Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

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

5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

4	3	4
2	4	3
2	3	4

3x3 output/feature

Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

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

5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

4	3	4
2	4	3
2	3	4

3x3 output/feature

Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

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

5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

4	3	4
2	4	3
2	3	4

3x3 output/feature

Convolutional layer

The convolutional layer **convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”.

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

5x5 image

1	0	1
0	1	0
1	0	1

3x3 filter/kernel

4	3	4
2	4	3
2	3	4

3x3 output/feature

Padding

When kernel size and stride cannot fit the image dimension, add 0s outside the boundaries.

padding = 1

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

7x7 image

output size = (N-F)/stride + 1

1	0	1
0	1	0
1	0	1

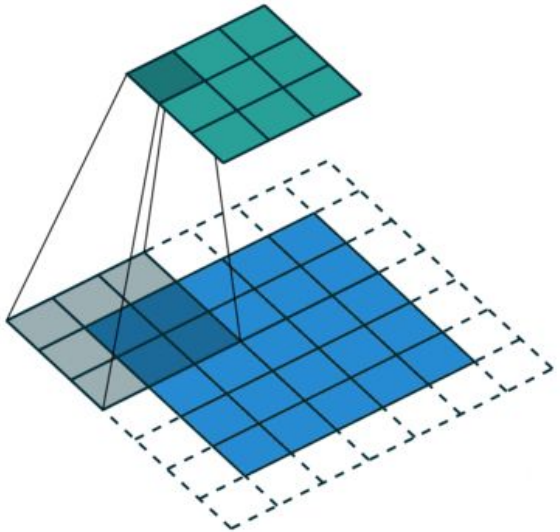
3x3 filter/kernel

2	2	0
1	2	1
0	2	2

3x3 output/feature

PyTorch implementation

torch.nn.Conv2d(in_channels=1, out_channels=1, kernel_size=(3,3), stride=(2,2), padding=1)

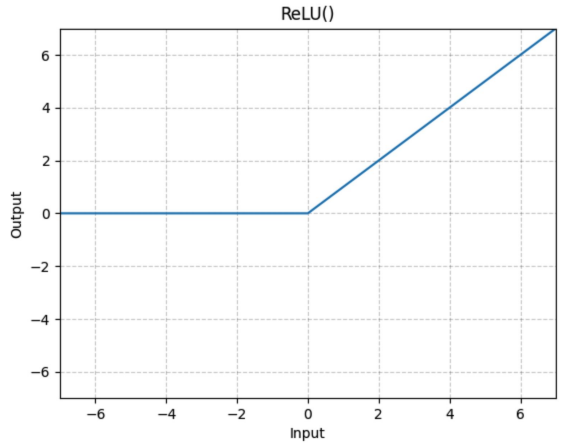


Note: default format for image data in PyTorch: NCHW

Activation function: ReLU

Adding nonlinearity to the neural network

Pytorch implementation: **torch.nn.ReLU()**



$$\mathrm{ReLU}(x) = (x)^+ = \max(0, x)$$

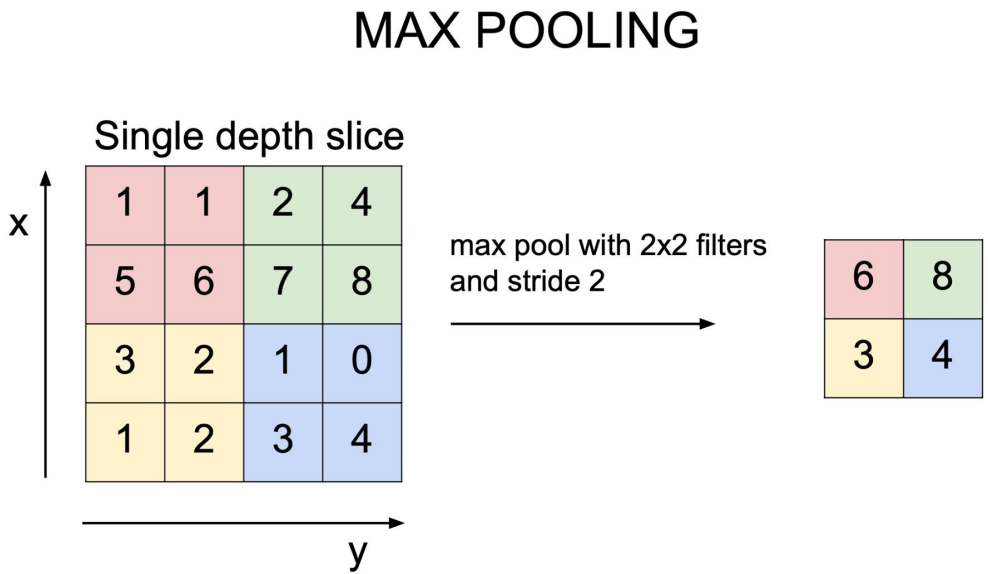
Pooling layer

The pooling layer makes the representations smaller and more manageable..

PyTorch implementation: **torch.nn.MaxPool2d(kernel_size=2)**

Other pooling layer APIs: https://pytorch.org/docs/stable/nn.html#pooling-layers

- max pooling
- min pooling
- average pooling



Fully connected layer

In the fully-connected layer, each node in the output layer connects directly to a node in the previous layer.

Pytorch implementation: **torch.nn.Linear(input_size, output_size)**