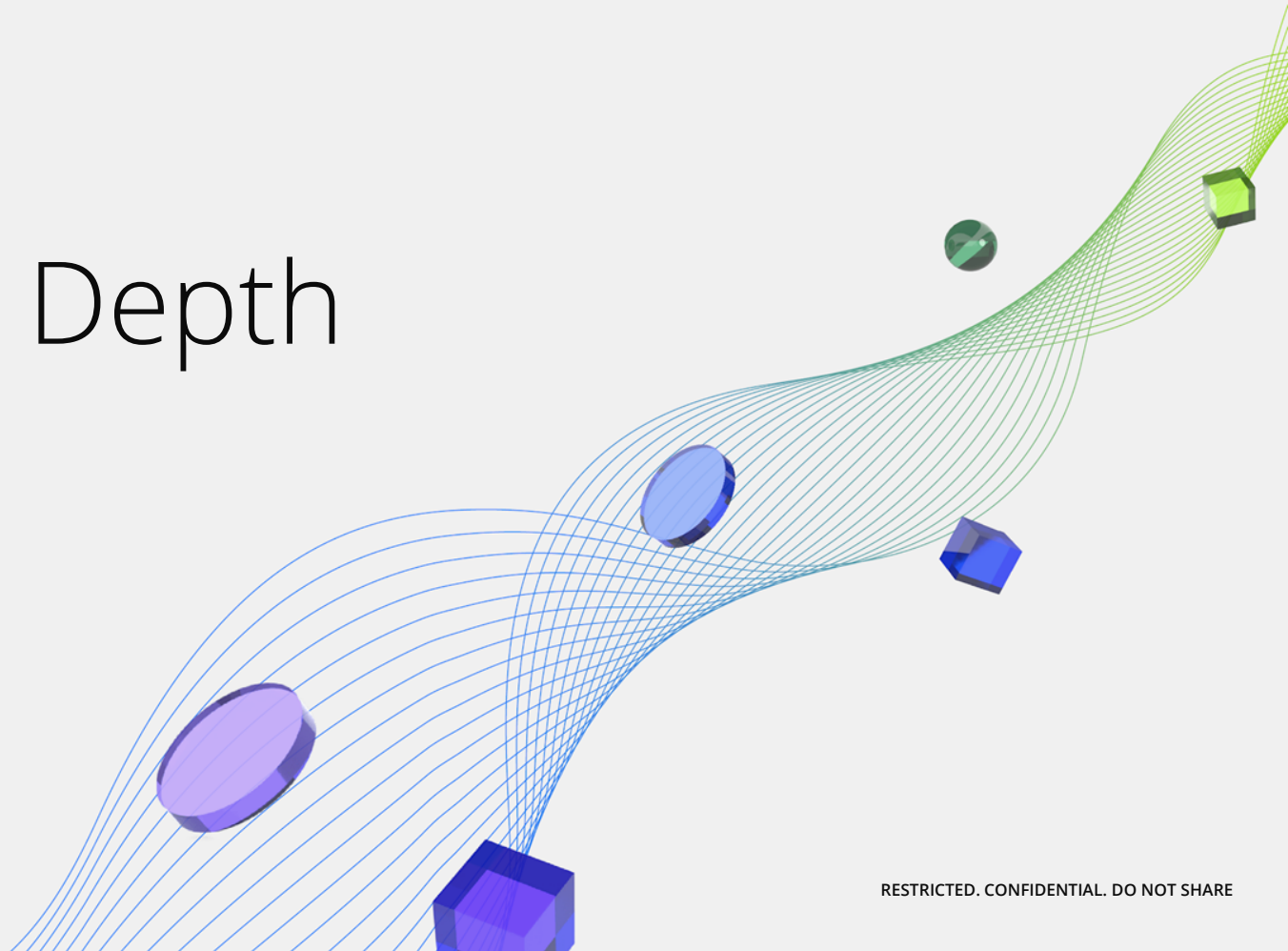
UDACITY

CNNs In Depth

Lamia Zain
lamiahasan4@gmail.com
LinkedIn

June,21, 2024



Connect Sessions | Purpose

A Connect Session **IS**:

- Focused on learning, encouragement & graduation for a group of students coached by a Udacity Session Lead
- Setting weekly study goals
- Helping each other with progress (including peer to peer)
- Keeping everyone accountable for their responsibilities
- A way to meet individuals in tech field & learn about the industry
- **Mandatory**

A Connect Session **IS NOT**:

- A social meetup
- A study group
- A substitute for online learning
- **Optional**

UDACITY

Let’s check your progress

You are encouraged to spend at lest 10 hours/week to graduate.

Presentation date

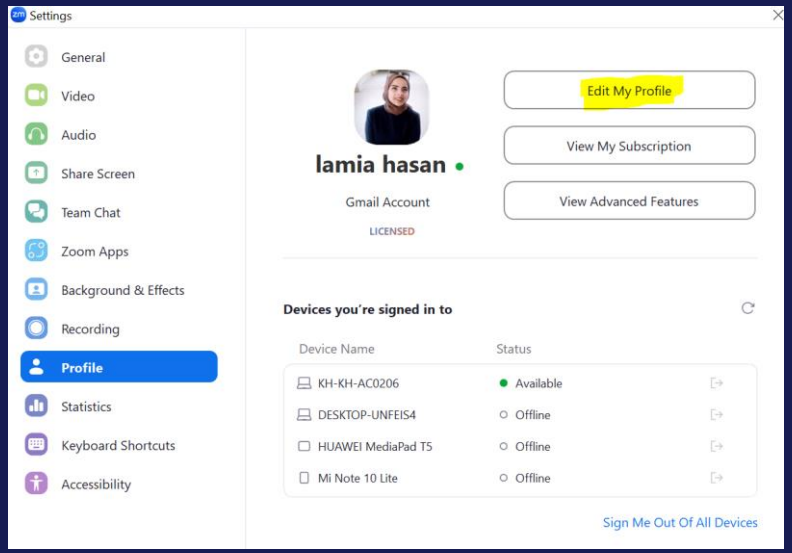
UDACITY

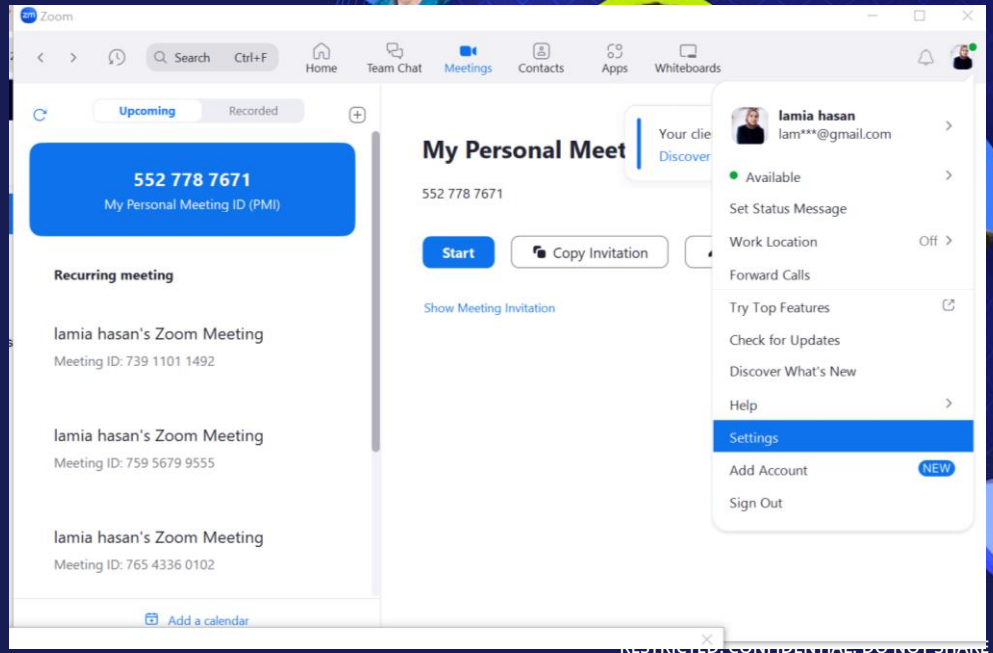
Attendance is taken automatically

Please change your name to be First Name and Last name on Zoom
Like : Lamia Zain

UDACITY

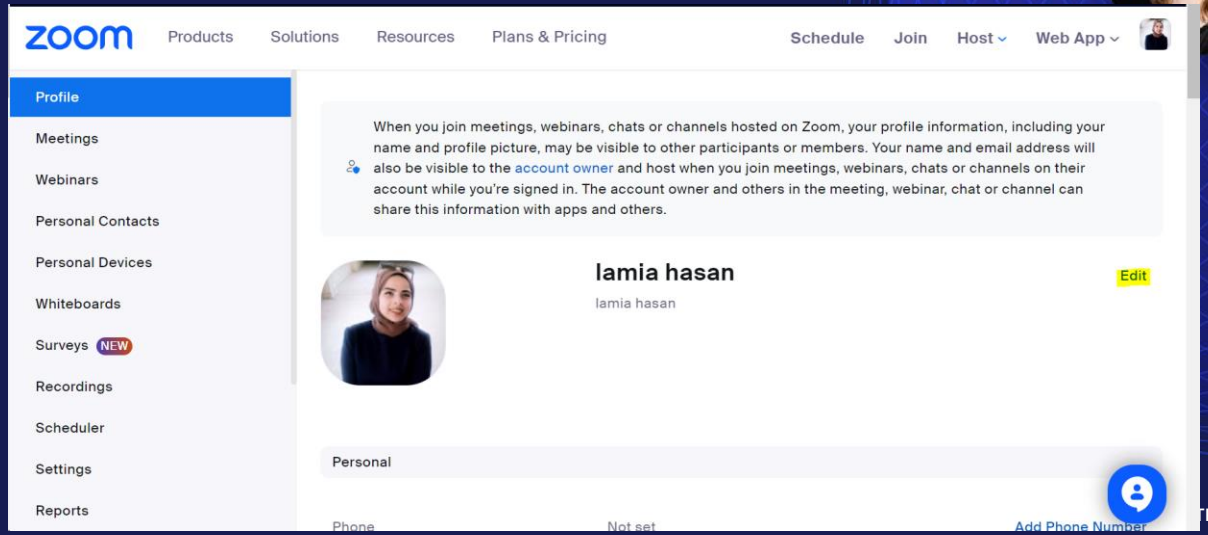
Change your Name on Zoom





UDACITY

Change your Name on Zoom



Session Lead role:

Communication Chart

Issue	Where to go?
Classroom access/ Withdrawal/ Graduation issues/ Plagiarism/ Project Review Inquiries	Email support@udacity.com
Technical Issues, Attendance, Content Related Issues/ Project inquiries	Session Lead
Session Switch/ Community related issues	Community Moderators

Program Timeline & Learning Journey

2024

April | May | June | July | August

- **Program Kickoff** — April 10, 2024
- **Revoking Deadline** — May 12, 2024
- **First Project Submission** — May 1, 2024
- **Second Project Submission** — June 5, 2024
- **Third Project Submission** — July 10, 2024
- **Fourth Project Submission** — August 21, 2024
- **End of Program** — August 28, 2024

Program Period — April 10 - August 28, 2024

Four-weeks Agenda, Weekly schedule

Week 10	Jun 12, 2024			Finish the lessons below from **the Convolutional Neural Networks** Introduction to CNNs CNN Concepts **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Introduction to CNNs CNN Concepts
Week 11	Jun 19, 2024			Finish the lessons below from **the Convolutional Neural Networks** CNNs in Depth **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks CNNs in Depth
Week 12	Jun 26, 2024			Finish the lessons below from **the Convolutional Neural Networks** Transfer Learning **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Transfer Learning
Week 13	Jul 3, 2024			Finish the lessons below from **the Convolutional Neural Networks** Autoencoders **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Autoencoders Project Walkthrough: Landmark Classification & Tagging for Social Media
Week 14	Jul 10, 2024	Jul 10, 2024	**Landmark Classification & Tagging for Social Media**	Finish the lessons below from **the Convolutional Neural Networks** Object Detection and Segmentation **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Object Detection and Segmentation Project Walkthrough: Landmark Classification & Tagging for Social Media

Student Milestone | Revoking

REVOKING

Revoking is the process by which Udacity removes a student from a Nanodegree program.

AWS reserves the right to revoke you from the program if you do not comply with program requirements.

CRITERIA

Students can be revoked if they fail to:

- Submit Project 1
- Complete the required concepts

The date of **Revoking** for the scholarship is **May 12, 2024**

Code of Conduct | Plagiarism

BASIC RULES

- Project submissions must consist of original work
- Submitted projects will be scanned for plagiarism
- Students who are found to have plagiarised will risk their Nanodegree being revoked
- Read the honor code and the rubric carefully for all projects

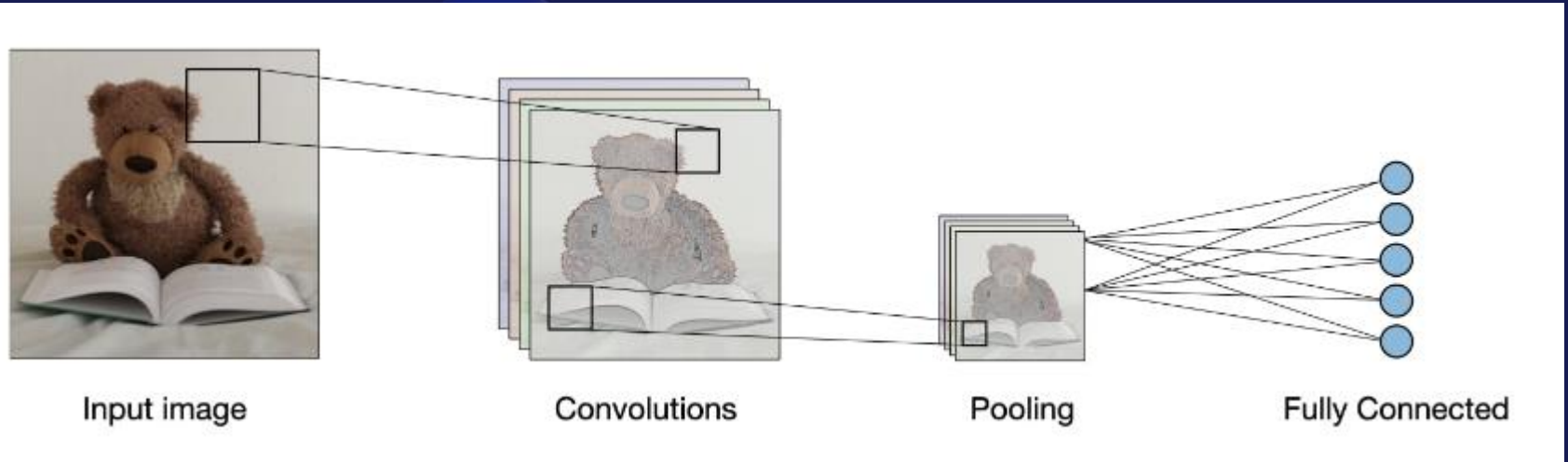
Recap

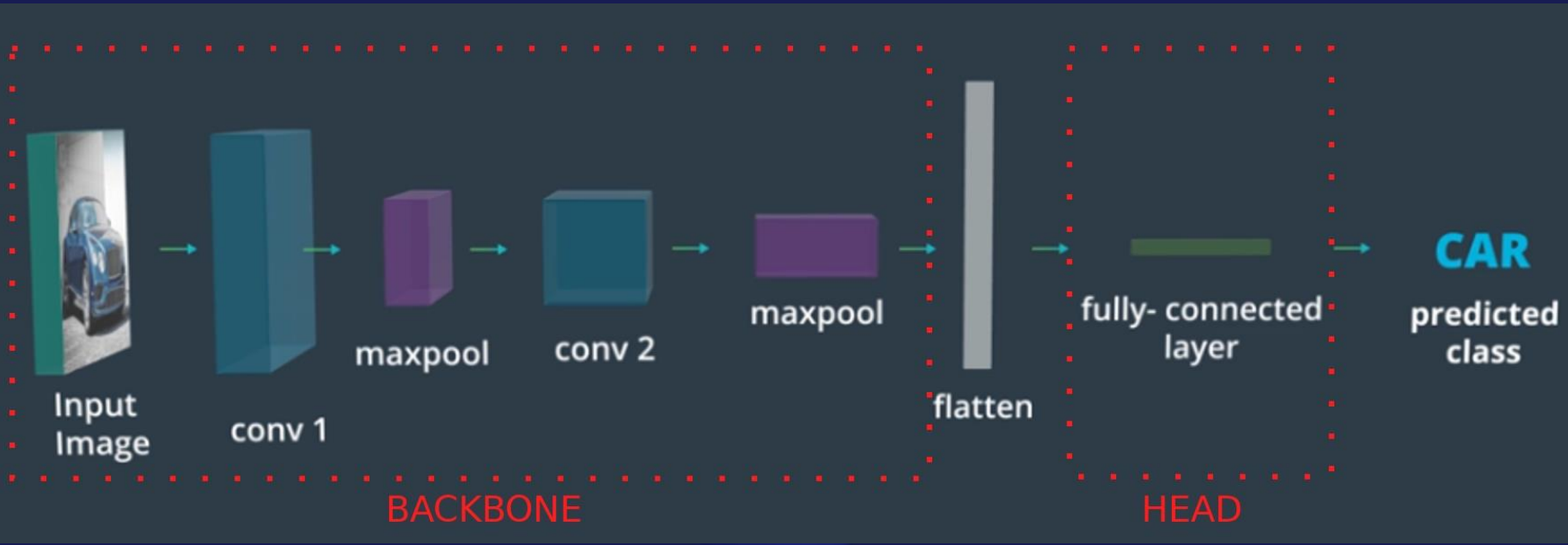
The difference between CNNS and multi perceptron networks when classifying Images

	CNNs	MLPs
Architecture	• Convolutional layers that apply filters to the input image. Capturing local spatial patterns. • followed by pooling layers that downsample the feature maps	• MLPs consist of fully connected layers where each neuron is connected to every neuron in the previous and subsequent layers • MLPs do not take into account the spatial structure of the input data. • Too Unnecessary Many Parameters
Parameter Sharing	• Use **parameter sharing, i.e,** the same filter is applied to different regions of the input image. • Smaller number of learned parameters compared to MLPs • Hence better for huge input data like images	• Each neuron is trained independently • Network learns unique weights for every connection. • Large number of parameters. • Computationally expensive for image data.
Translation Invariance	Due to their use of convolutional and pooling layers, they can detect local patterns regardless of their position in the image.	Sensitive to the position of pixels in the input image, making them less robust to translation or slight changes in input position.

CNN Architecture

Cs230 – Stanford University





Convolution Layer

- Responsible for preserving special information about an image.
- Detects features of images.
- By applying learnable Kernels/filters (Wights in MLPs).
- To produce feature maps.
- Size of the output feature maps is dependent on the padding and stride .

Example

- Input image of size 8x8.
- Kernel size of 3x3 stride of 1
- Padding= “valid”

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

Kernel

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2

Image

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
0.1	0.2	0.3	0.4	0.5	0.4	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.6	0.3	0.2
0.1	0.2	0.3	0.4	0.5	0.4	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.6	0.3	0.2
0.1	0.2	0.3	0.4	0.5	0.4	0.7	0.8
0.9	0.8	0.7	0.6	0.5	0.6	0.3	0.2

Applying kernels

0	0	0			

0	0	0	-0.2	4.6	4.9
-0.2	-0.2	-0.2	-0.4	4.2	3.9
0.2	0.2	0.2	0	4.6	4.9
0	0	0	0	4.4	4.1
0	0	0	0	4.6	4.9
0	0	0	0	4.4	4.1

Final Image

Feature map Size

$$Output\ size\ = \frac{Input\ size\ -\ Filter\ size\ + (\text{Number of Bads in a direction})}{\text{Stride}} + 1$$

- Input size: The spatial dimensions (width or height) of the input Image.
- Filter size: The spatial dimensions (width or height) of the convolutional filter (kernel).
- Padding: The number of pixels added to the borders of the input feature map before applying the convolution. Padding helps preserve spatial dimensions and information.
- Stride: The number of steps the filter moves (horizontally or vertically) as it slides over the input feature map during the convolution operation.
- For the previous example,

$Output\ Height = \frac{8-3}{1} + 1 = 6$

$Output\ width\ = \frac{8-3}{1} + 1 = 6$

- It's important to note that for a 2D convolution, the formula applies independently to both the width and height dimensions of the input.

Convolution Layer

- Responsible for preserving special information about an image.
- Detects features of images.
- By applying learnable Kernels/filters (Wights in MLPs).
- To produce feature maps.
- Size of the output feature maps is dependent on the padding and stride .

Example

- Input image of size 8x8.
- Kernel size of 3x3 stride of 1
- Padding= “Same”
- Padding_mode=‘zeros’

Convolution Process

- Center the filter
- Element-wise Multiplication.
- Summation
- Replace Pixel value
- Shift and Repeat:

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

Kernel

0	0	0	0	0	0	0	0	0	0
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0
0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0
0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0
0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0
0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0
0	0	0	0	0	0	0	0	0	0

2D image

1.2	0.2	0.2	0.2	0.2	0.2	0.2	-1.7
2	0	0	0	-0.2	0.2	0.2	-2
2	0	0	0	-0.2	0	0.2	-2
2	0	0	0	0	0	0	-2
2	0	0	0	0	0	0	-2
2	0	0	0	0	0	0	-2
2	0	0	0	0	0	0	-2
1.8	-0.2	-0.2	-0.2	0	-0.2	-0.4	-1.3

Output Feature Map

Feature map Size

$$Output\ size\ = \frac{Input\ size\ -\ Filter\ size\ + (\text{Number of Bads in a direction})}{\text{Stride}} + 1$$

- Input size: The spatial dimensions (width or height) of the input Image.
- Filter size: The spatial dimensions (width or height) of the convolutional filter (kernel).
- Padding: The number of pixels added to the borders of the input feature map before applying the convolution. Padding helps preserve spatial dimensions and information.
- Stride: The number of steps the filter moves (horizontally or vertically) as it slides over the input feature map during the convolution operation.
- For the previous example,

$Output\ Height = \frac{8-3+2}{1} + 1 = 8$

$Output\ width\ = \frac{8-3+2}{1} + 1 = 8$

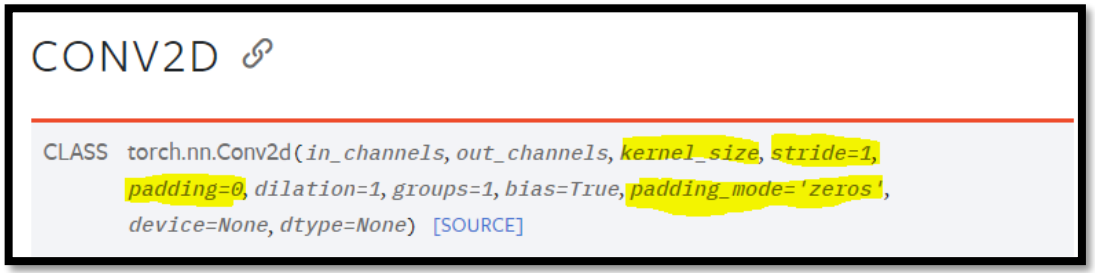
- It's important to note that for a 2D convolution, the formula applies independently to both the width and height dimensions of the input.

Stride:

- Number of steps the convolutional filter (kernel) takes as it slides over the input data (Stride=1)

Padding:

- Adding extra pixels around the borders of the input data before applying the convolution operation
- Without padding, the size of the feature maps reduces with each convolution operation.
- which can lead to a progressive loss of spatial information
- In PyTorch Padding =
- "valid" (no padding),
- "same" (padding to keep output size **same** as input size **with stride 1.**



Conv2d in Pytorch (captures spatial hierarchies in the data)

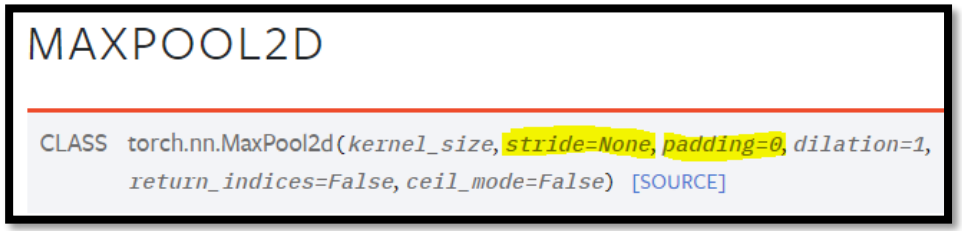
Conv1d in Pytorch (captures temporal or sequential patterns in the data)

Pooling:

- Downsamples the spatial dimensions of the feature maps obtained after convolutional layers.
- Reduces the number of parameters in the network.
- The two most common types of pooling layers are MaxPooling and AveragePooling

MaxPooling:

- The most common type of pooling used in CNNs.
- It operates on small regions (usually 2x2 or 3x3) of the input feature maps.
- Outputs the maximum value of the elements within that region.
- The stride parameter determines the step size at which the pooling window moves over the feature map.
- Typically, the stride is set to the same value as the pooling window size to avoid overlapping



If you have a feature map of size 8x8 and MaxPolling layer of size (2,2) what do you think the size of the output will be ??

Pooling:

- Downsamples the spatial dimensions of the feature maps obtained after convolutional layers.
- Reduces the number of parameters in the network.
- The two most common types of pooling layers are MaxPooling and AveragePooling

MaxPooling:

1.2	0.2	0.2	0.2	0.2	0.2	0.2	-1.7
2	0	0	0	-0.2	0.2	0.2	-2
2	0	0	0	-0.2	0	0.2	-2
2	0	0	0	0	0	0	-2
2	0	0	0	0	0	0	-2
2	0	0	0	0	0	0	-2
2	0	0	0	0	0	0	-2
1.8	-0.2	-0.2	-0.2	0	-0.2	-0.4	-1.3

Output Feature Map Before Booling

2	0.2	0.2	0.2
2	0	0	0.2
2	0	0	0
1.8	0	0	0

After Pooling

Pooling:

- Downsamples the spatial dimensions of the feature maps obtained after convolutional layers.
- Reduces the number of parameters in the network.
- The two most common types of pooling layers are MaxPooling and AveragePooling

AveragePooling:

- Less Commonly used
- Computes the average of the elements within each pooling window instead of taking the maximum.

```
CLASS  torch.nn.AvgPool2d(kernel_size, stride=None, padding=0,
       ceil_mode=False, count_include_pad=True, divisor_override=None)
```

Break (10 minutes)

Satisfaction Survey

Train a Simple CNN for Fashion MNIST Dataset

Why Do you think CNN Gives Less Accuracy than MLPs

Any Question?

Thank you

UDACITY