Deep Learning

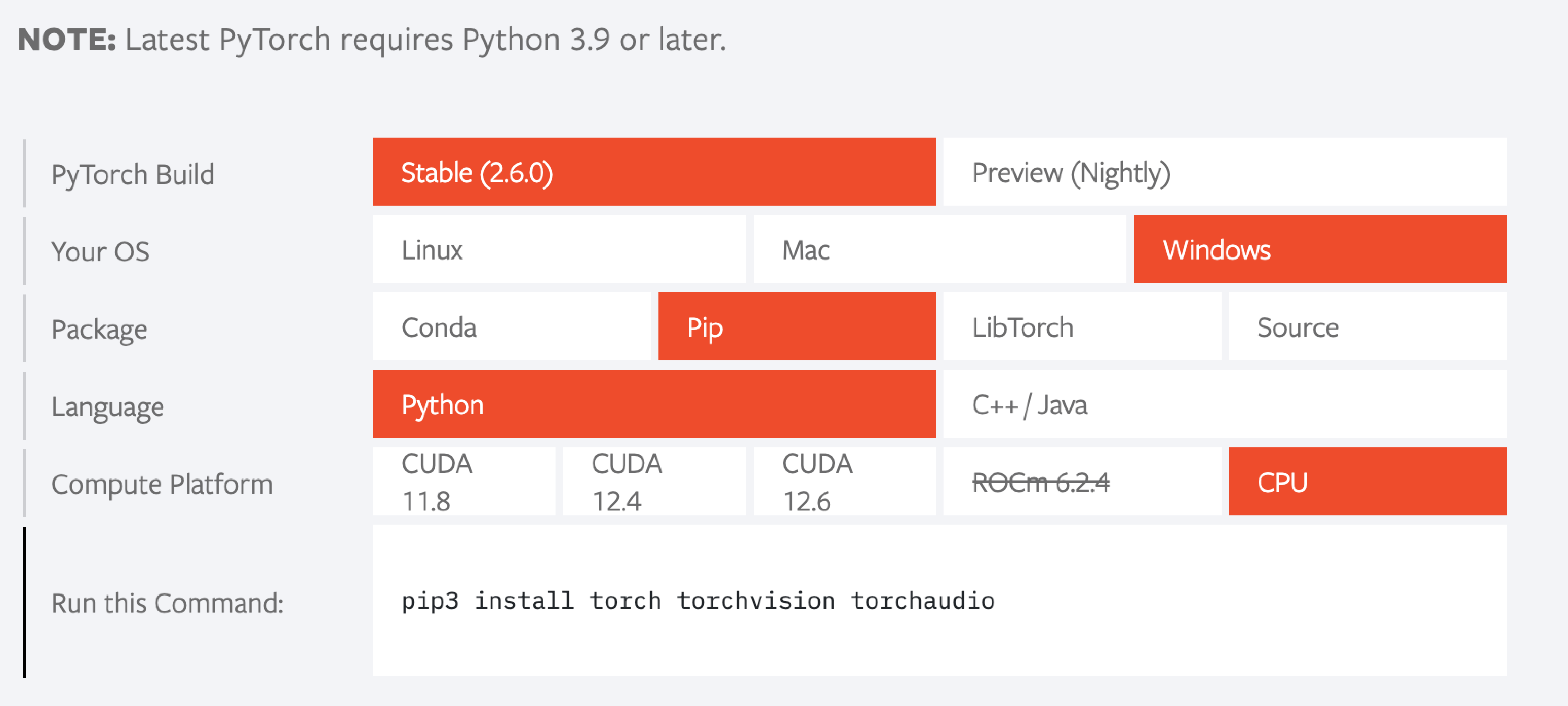
Yao Zhang - Aalto University

Expected Outcomes

- How to create MLP/CNN-LSTM model
- How to train a deep learning model by Pytorch

Pytorch

```
https://pytorch.org/
```



NOTE: Latest PyTorch requires Python 3.9 or later.

PyTorch Build	[✓ Stable (2.6.0)]			Preview (Nightly)	
Your OS	Linux		Mac		[✓ Windows]
Package	Conda	[✓ Pip]		LibTorch	Source
Language	[✓ Python]			C++ / Java	
Compute Platform	CUDA 11.8	CUDA 12.4	CUDA 12.6	~~ROCm 6.2.4~~	[✓ CPU]

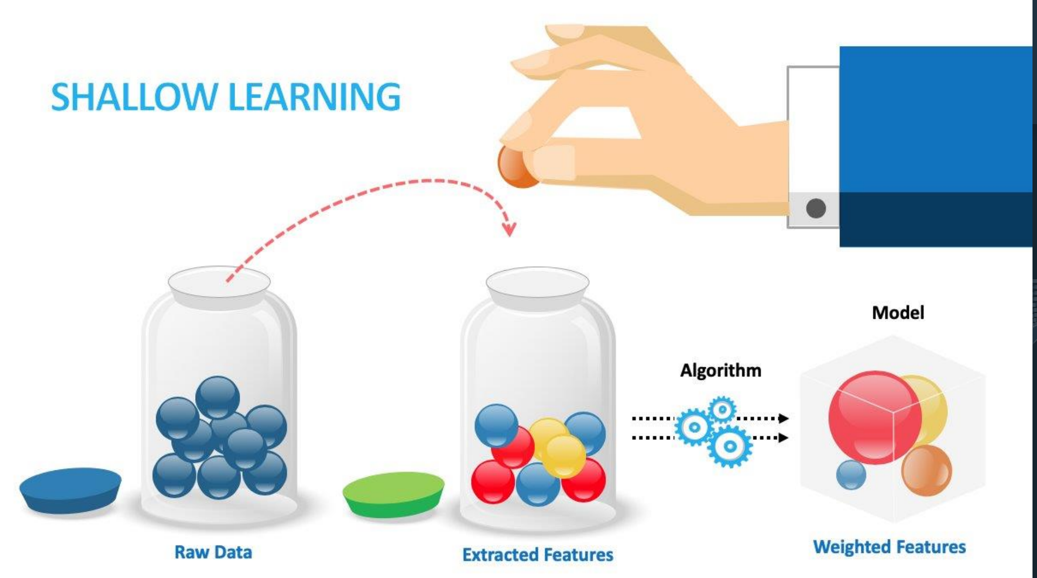
Run this Command:

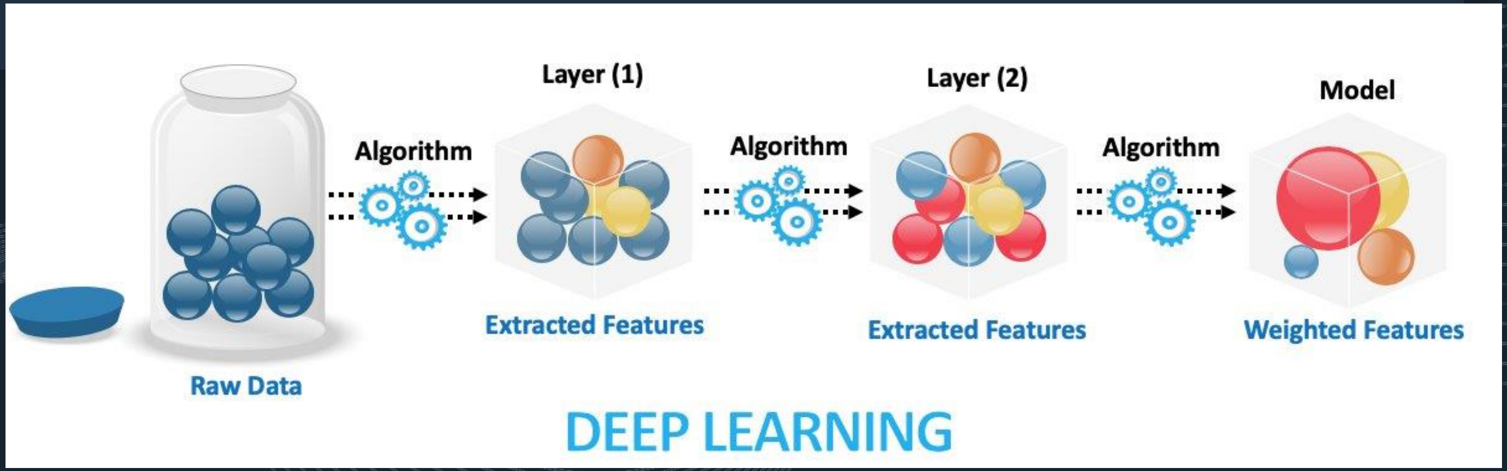
```
pip3 install torch torchvision torchaudio
```

differences

Shallow Learning: we extract features from the channels across the entire sliding window. So each sliding window is transformed into 2 features (for example, the mean total acceleration and the mean total angular speed)

Deep Learning: we do not extract features but give the whole data of the sliding window at once. That is, 32*2 (64) values. The deep neural network is supposed to learn features from the "raw" data. And deep learning model also have a much higher parameter and complexity

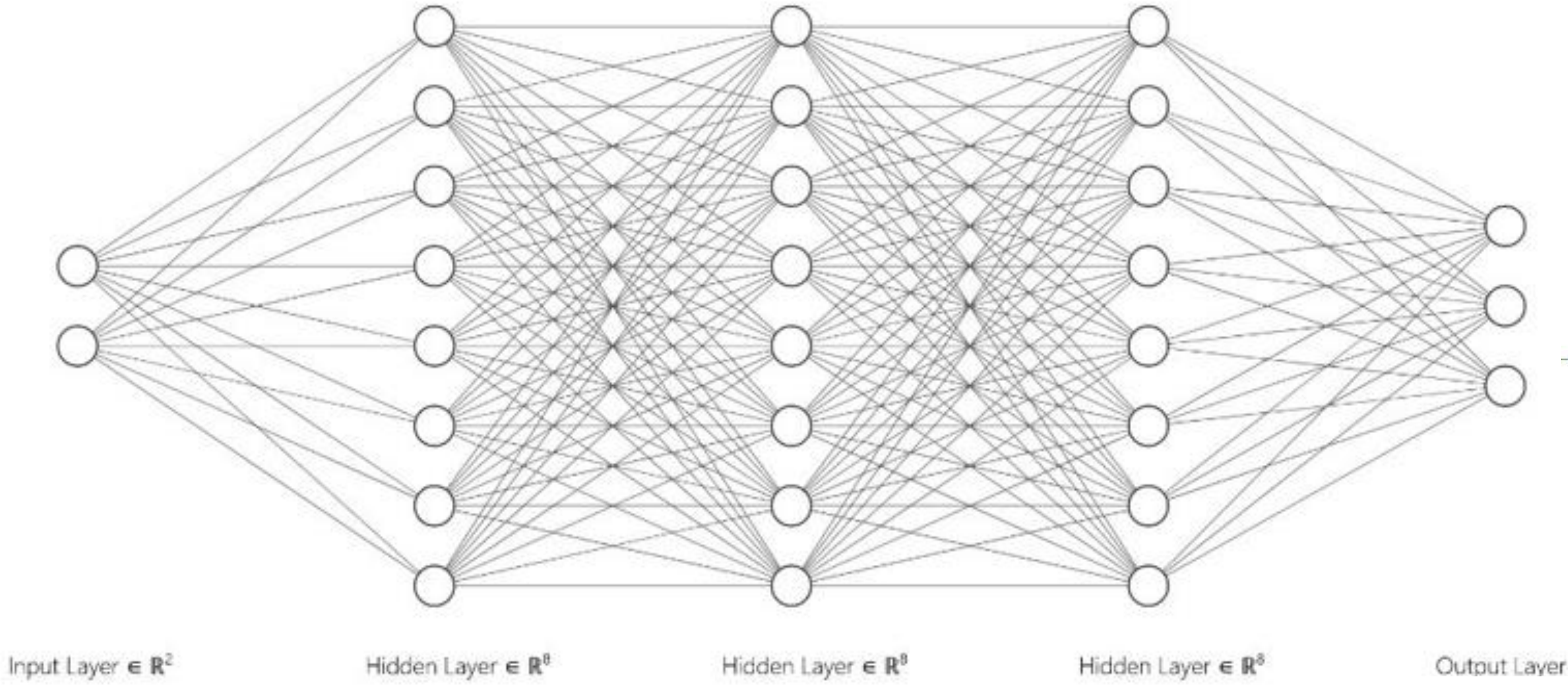


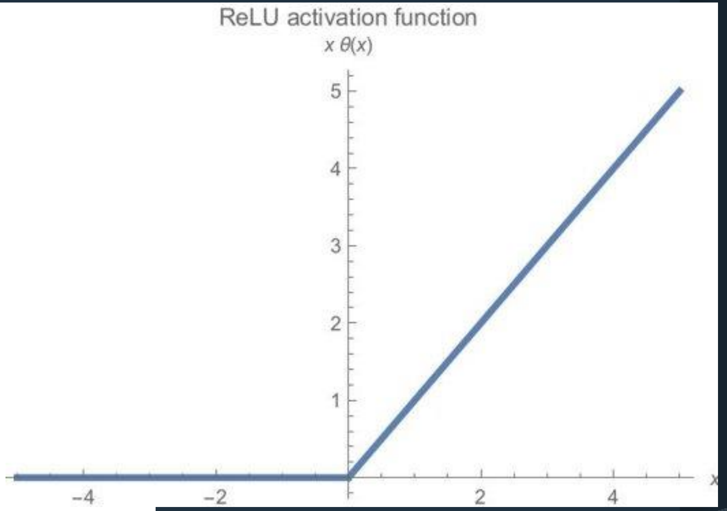


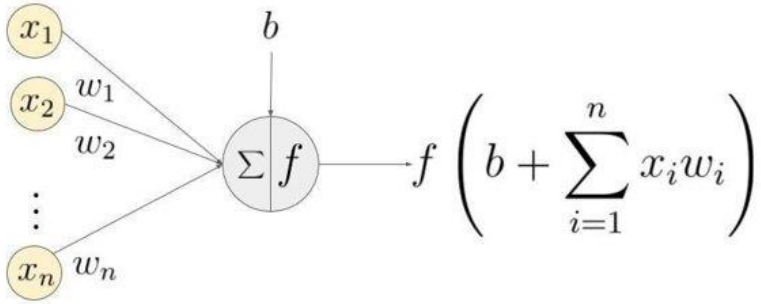
Steps for training Deep learning model

- **STEP 1.** We define the number of layers and their neurons.
- **STEP 2.** We initialize the weights and biases.
- **STEP 3.** We choose an optimizer, loss function, and learning rate. These three things will work together to gradually change the weights.
- **STEP 4.** We start the learning (training or optimization).
- **STEP 5.** The training ends according to certain criteria

Multi-layer perceptron (MLP)

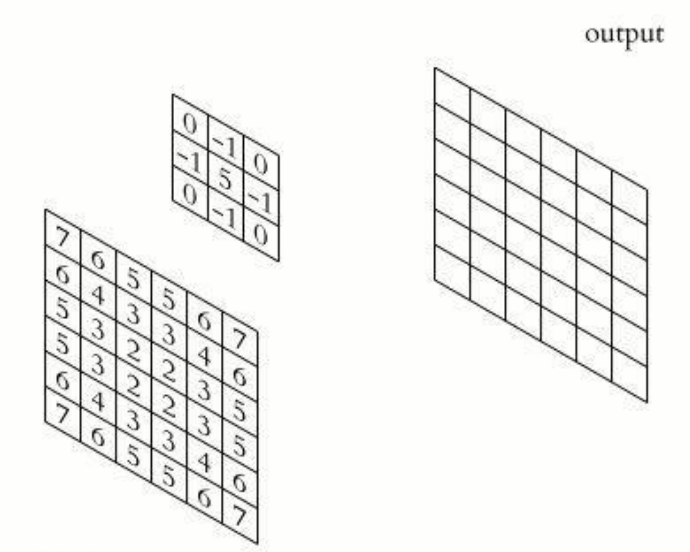

Input Layer $\in \mathbb{R}^2$
Hidden Layer $\in \mathbb{R}^8$
Hidden Layer $\in \mathbb{R}^8$
Hidden Layer $\in \mathbb{R}^8$
Output Layer $\in \mathbb{R}^3$





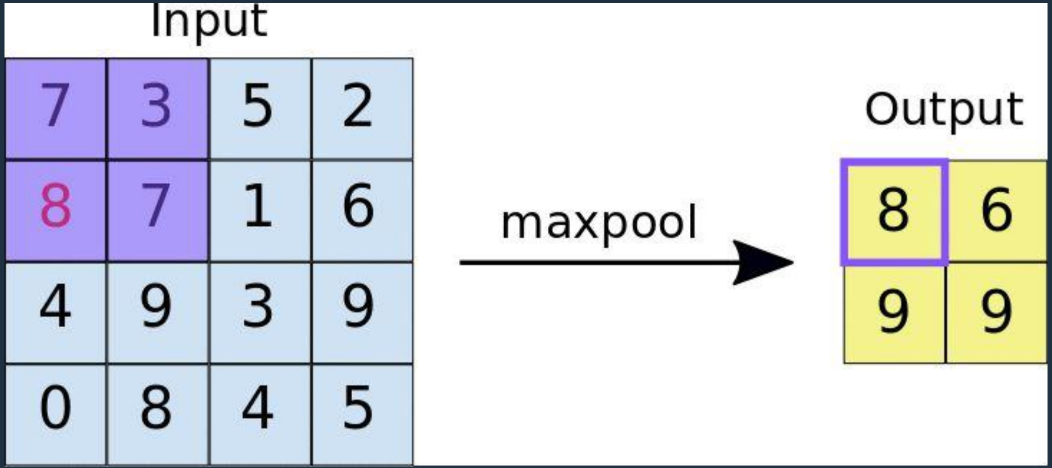
An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Convolutional Neural Network (CNN)



0	-1	0
-1	5	-1
0	-1	0

7	6	5	5	6	7
6	4	3	3	4	6
5	3	2	2	3	5
5	3	2	2	3	5
6	4	3	3	4	6
7	6	5	5	6	7

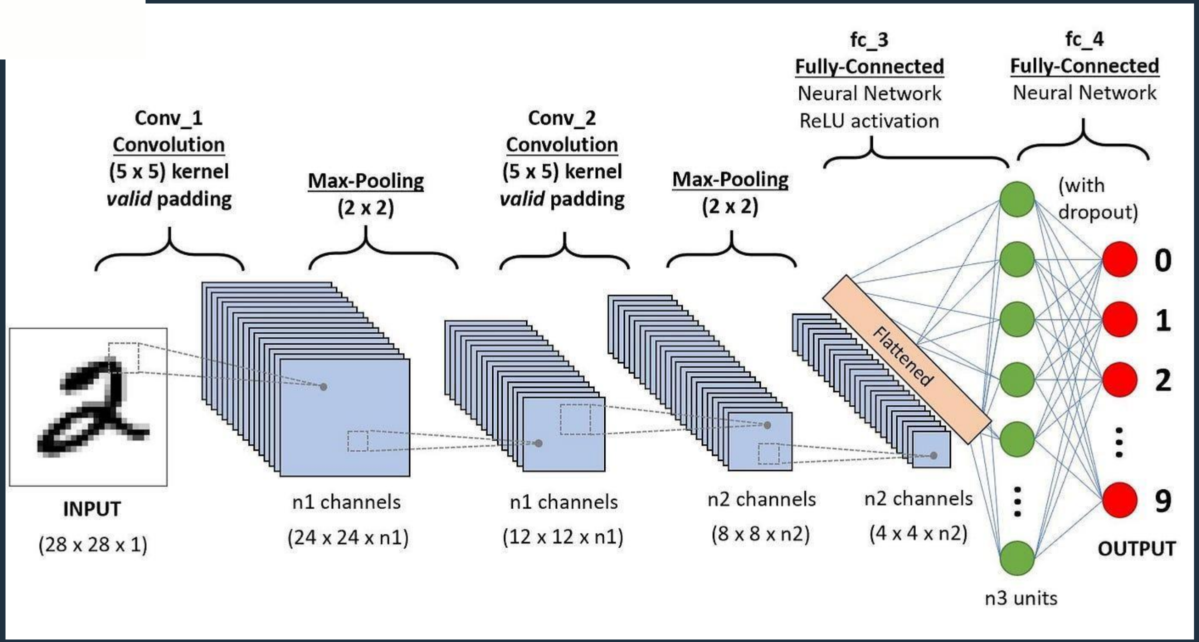


Input

7	3	5	2
8	7	1	6
4	9	3	9
0	8	4	5

Output

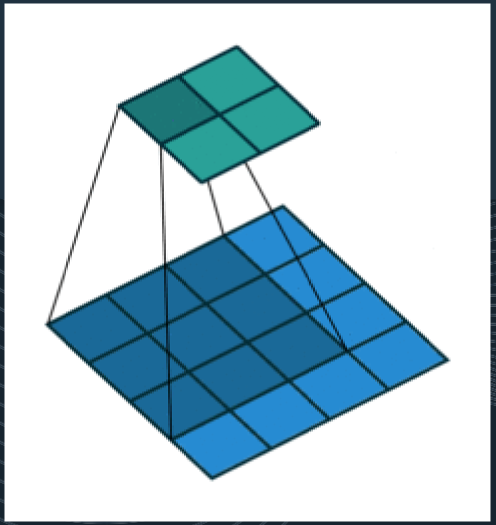
8	6
9	9



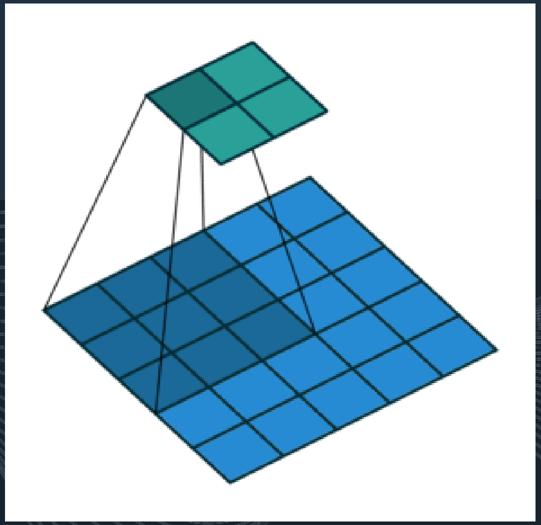
Convolutional Neural Network (CNN)

Kenel size = (3,3)

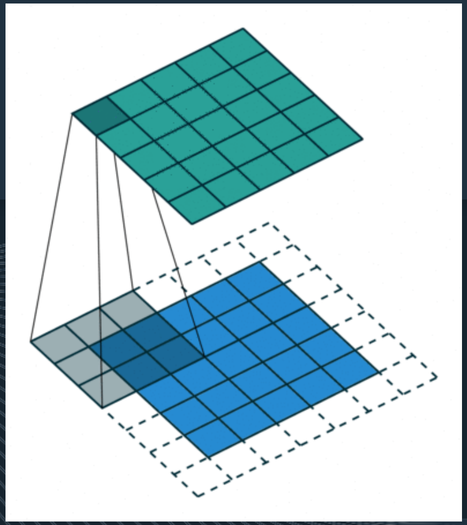
Stride=(1,1)
Padding=0



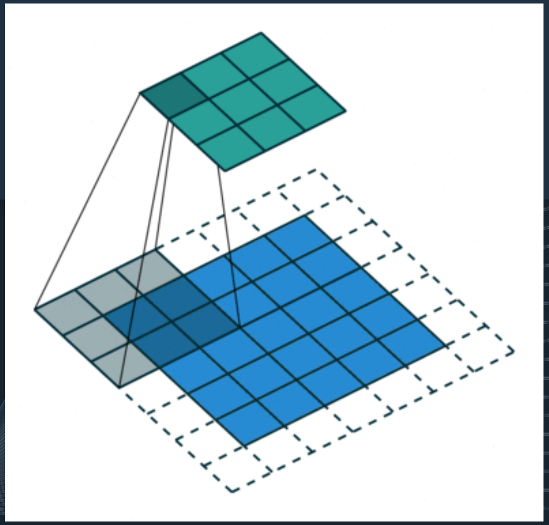
Stride=(2,2)
Padding=0



Stride=(1,1)
Padding=1



Stride=(2,1)
Padding=1



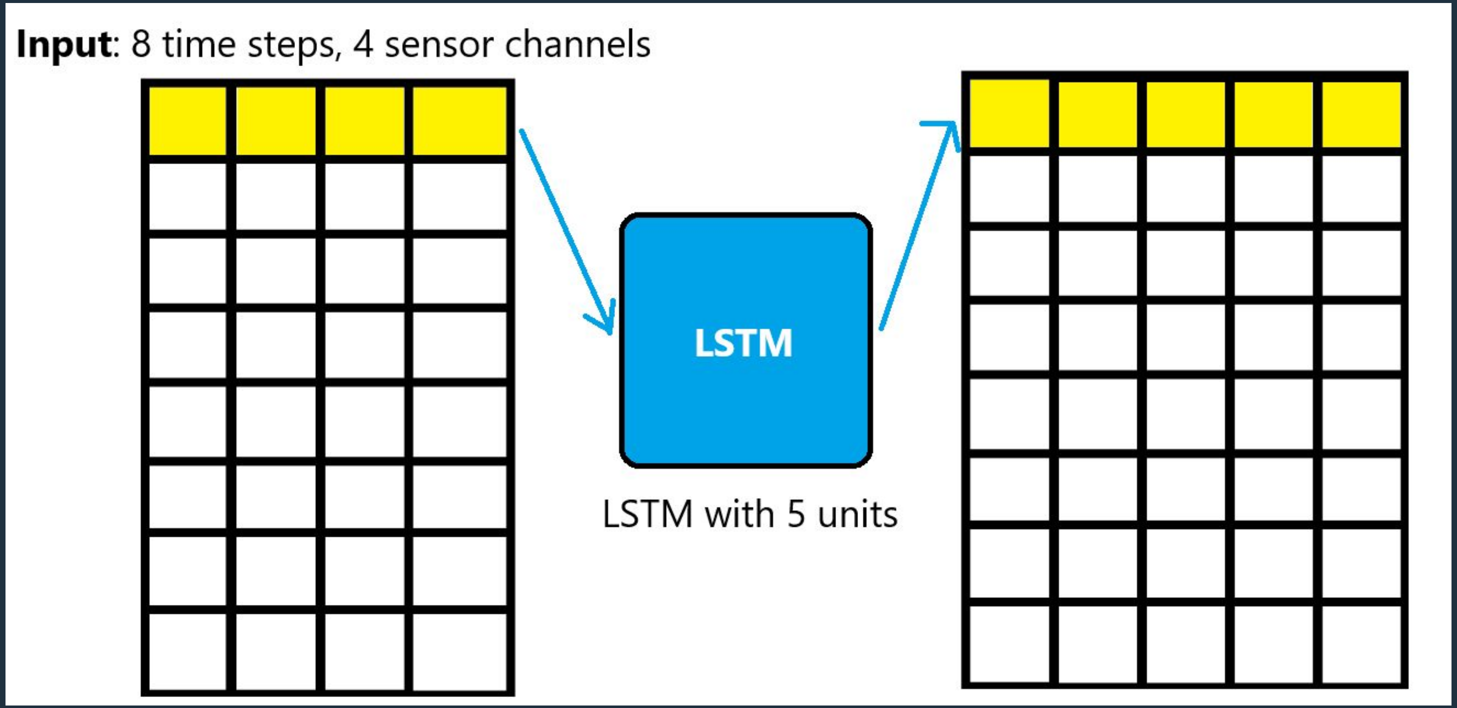
Deep Learning

- LSTMs (Long Short-Term Memory): for temporal relationship
- An LSTM layer processes the input (time-series data) step by step. Step *n* depends on all previous steps.

Input size:(batch_size, 8, 4)

Input gate, forget gate, output gate

Output size: (batch_size, 5), the last hidden state from LSTM(batch_size, 8, 5)



Summary

1. How to receive data from Arduino

2. How to pre-processing data

3. Different classification model: shallow learning methods

4. Model deployment: real-time prediction

5. Deep learning

Next lecture and tutorial : Q&A