Introduction to PyTorch

Joshua Yao-Yu Lin (林曜宇)
University of Illinois at Urbana-Champaign

[@HAL training 2021.9.29]

Center for Artificial Intelligence Innovation
NATIONAL CENTER FOR SUPERCOMPUTING APPLICATIONS | NCSA

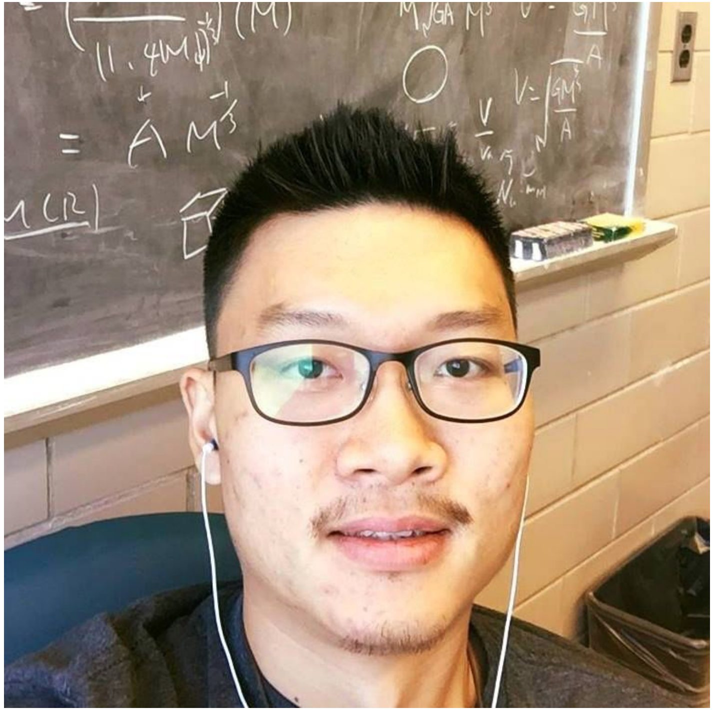
Joshua Yao-Yu Lin

- My research spans a wide range of Machine Learning application in Astrophysics
- Research Interest: dark matter, supermassive black holes, neuroscience, **machine learning**

- Before Joining UIUC, I got my MS at NTU, and BS at NTHU (All in physics).

- ML intern experience: Simons Foundation, Google Research

- I’ve used PyTorch for most of my deep learning projects!



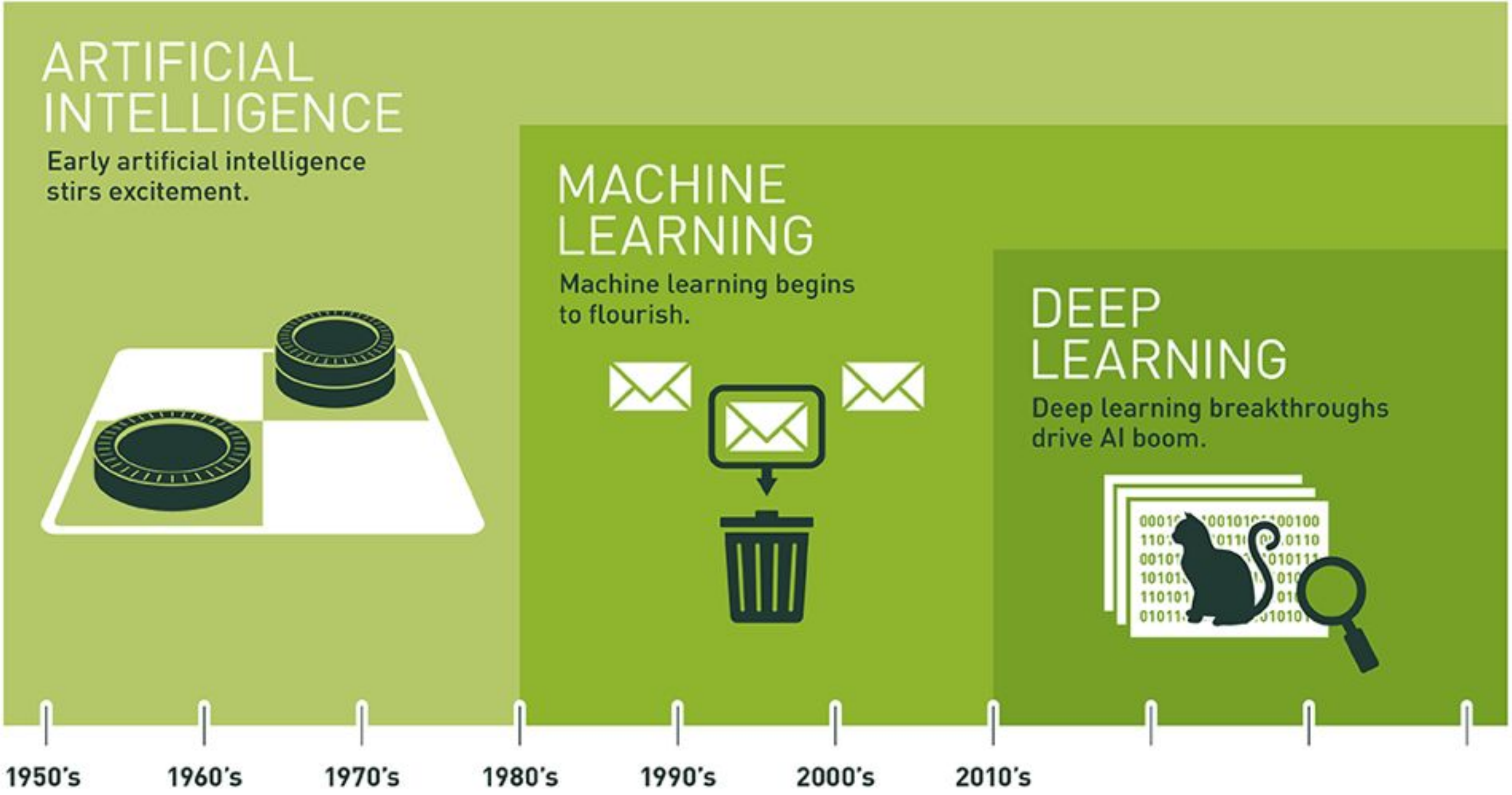
Physics PhD student (2016-Present)

University of Illinois at Urbana-Champaign

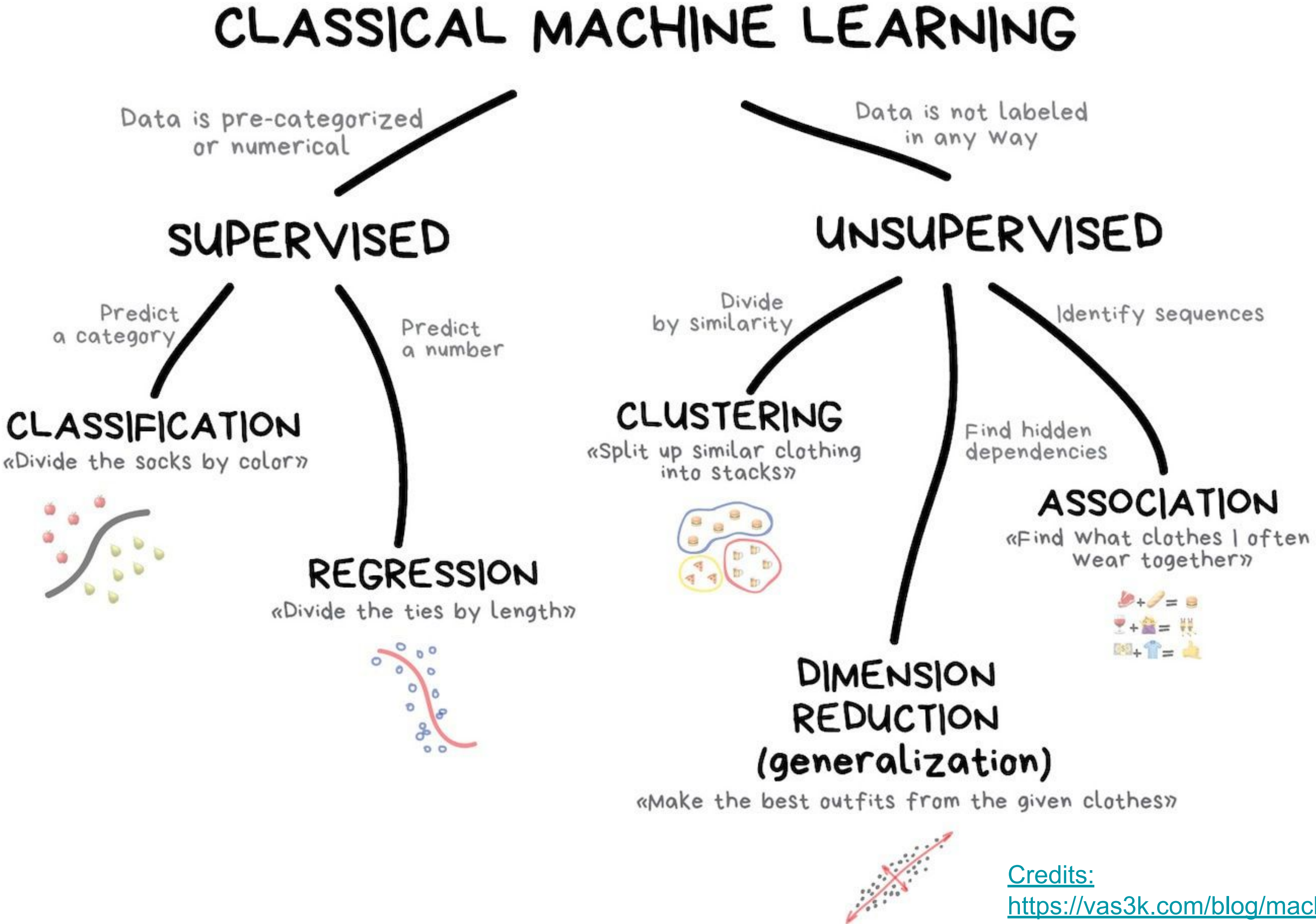
Agenda

- Overview of PyTorch & Deep Learning
- Pytorch Basics
- Train a Convolutional neural networks to classify MNIST data
- Train a Variational Autoencoder to generate new MNIST data
- Q & A section

https://github.com/joshualin24/NCSA_pytorch_tutorial



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.



Credits:
https://vas3k.com/blog/machine_learning/

What is PyTorch?

- Open source machine learning library
- Developed by Facebook's AI Research lab
- It leverages the power of GPUs
- Automatic computation of gradients
- Makes it easier to test and develop new ideas.

Why PyTorch?

TensorFlow

Keras

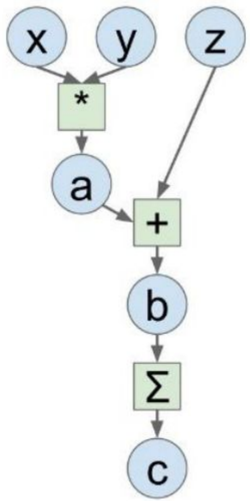
PyTorch

Why PyTorch?

- It is pythonic - concise, close to Python conventions
- Strong GPU support
- Autograd - automatic differentiation
- Many algorithms and components are already implemented
- Similar to NumPy

Why PyTorch?

Computation Graph



Numpy

```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```

Tensorflow

```
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

with tf.device('/gpu:0'):
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)
    z = tf.placeholder(tf.float32)

    a = x * y
    b = a + z
    c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

PyTorch

```
import torch

N, D = 3, 4

x = torch.rand((N, D),requires_grad=True)
y = torch.rand((N, D),requires_grad=True)
z = torch.rand((N, D),requires_grad=True)

a =x * y
b =a + z
c=torch.sum(b)

c.backward()
```

Pytorch Tensor

- Basic block of pytorch
- Similar to numpy array
- easy to operate on GPU/CPU
- Good for building neural networks

In [5]:

```
import torch

x= torch.tensor([[1,2,3],[4,5,6]])
y= torch.tensor([[7,8,9], [10,11,12]])

f= 2*x + y
print(f)
```

```
tensor([[  9,  12,  15],
        [ 18,  21,  24]])
```

Demo: pytorch basics

https://bit.ly/2XWflcL

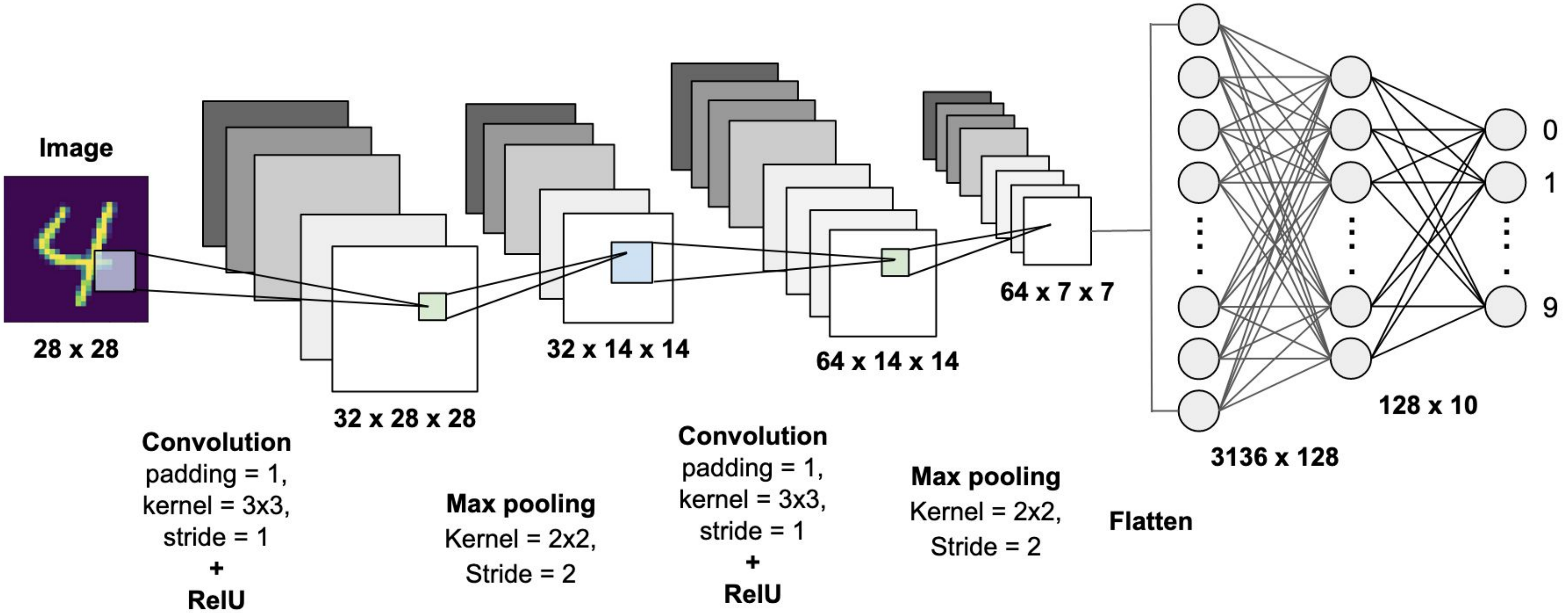
MNIST for image classification

MNIST in pixels

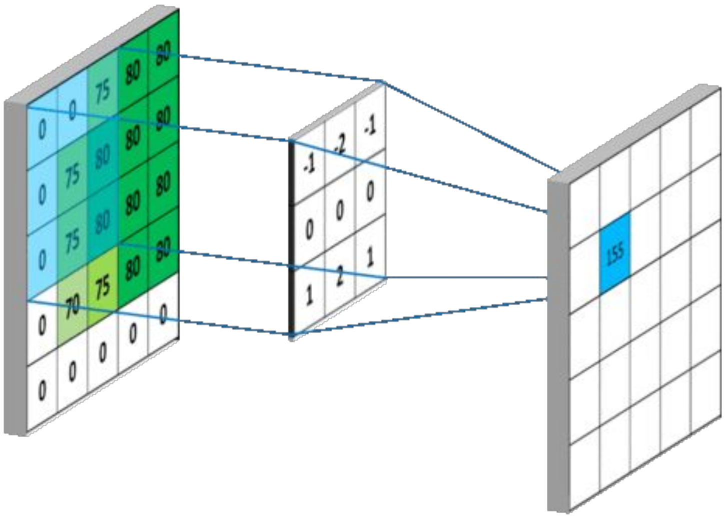
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	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
5	0	0	0	0	0	0	0	20	66	142	191	254	254	254	255	205	156	47	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	5	202	246	215	149	122	122	122	122	205	230	244	79	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	45	218	52	0	0	0	0	0	0	0	27	218	225	19	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	7	59	0	0	0	0	0	0	0	0	0	169	221	16	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	247	136	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	47	225	247	17	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	32	128	253	249	84	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	13	102	223	254	188	64	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	52	221	249	189	37	4	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	119	234	253	245	188	188	153	112	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	153	188	188	188	212	230	254	254	254	149	12	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	11	18	28	66	149	250	223	39	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	70	234	213	11	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	172	253	103	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	215	253	86	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	167	244	136	2	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	89	228	253	74	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	53	11	0	0	0	21	140	245	253	215	57	2	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	235	146	123	123	167	230	246	175	58	13	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	130	253	253	197	148	65	49	0	0	0	0	0	0	0	0	0	0	0	0	0
25	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
26	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
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	0	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	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	0	0	0	0	0
1	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
2	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
3	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
4	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
5	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
6	0	0	0	0	0	0	0	0	0	0	0	0	0	13	18	18	18	134	150	255	169	37	0	0	0	0	0	0
7	0	0	0	0	0	5	92	36	36	140	154	154	154	224	253	253	253	253	253	253	253	170	0	0	0	0	0	0
8	0	0	0	0	0	36	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	145	0	0	0	0	0	0
9	0	0	0	0	0	9	144	202	253	223	182	182	182	182	182	138	65	161	253	253	235	36	0	0	0	0	0	0
10	0	0	0	0	0	0	0	13	47	27	0	0	0	0	0	0	0	130	253	253	147	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	235	253	253	70	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	166	253	253	230	36	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	243	253	251	131	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	176	253	253	224	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	253	253	253	141	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	166	253	253	208	25	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	95	244	253	253	124	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	177	253	253	203	14	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	2	107	252	253	253	81	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	38	253	253	253	253	92	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	24	163	253	253	253	236	60	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	125	253	253	253	233	63	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	218	253	253	253	137	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	243	253	253	101	8	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	187	253	234	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	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
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	0	0	0	0

Introduction to neural networks with pytorch

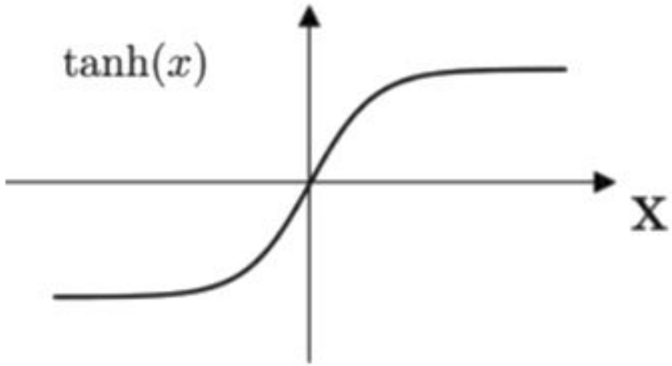


Convolution & Feature Map generation

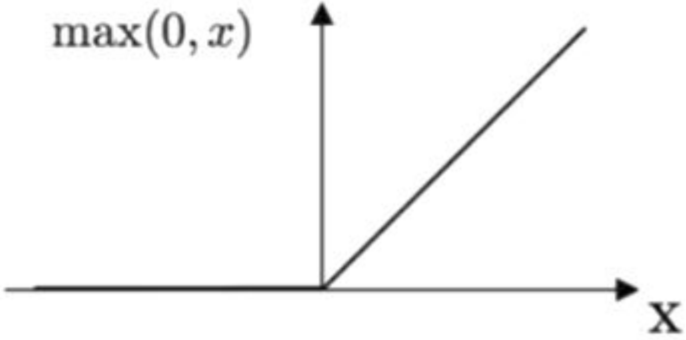


- Intermediate output insides the hidden layers of neural networks
- “Down sample”
- Single input can induce multiple feature maps (with different kernels)

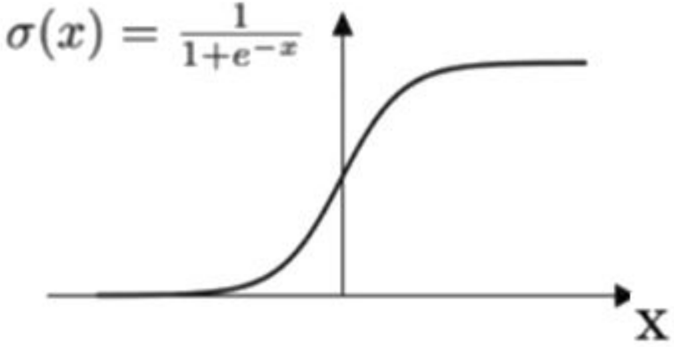
Tanh



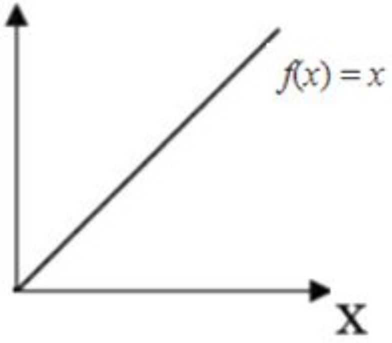
ReLU



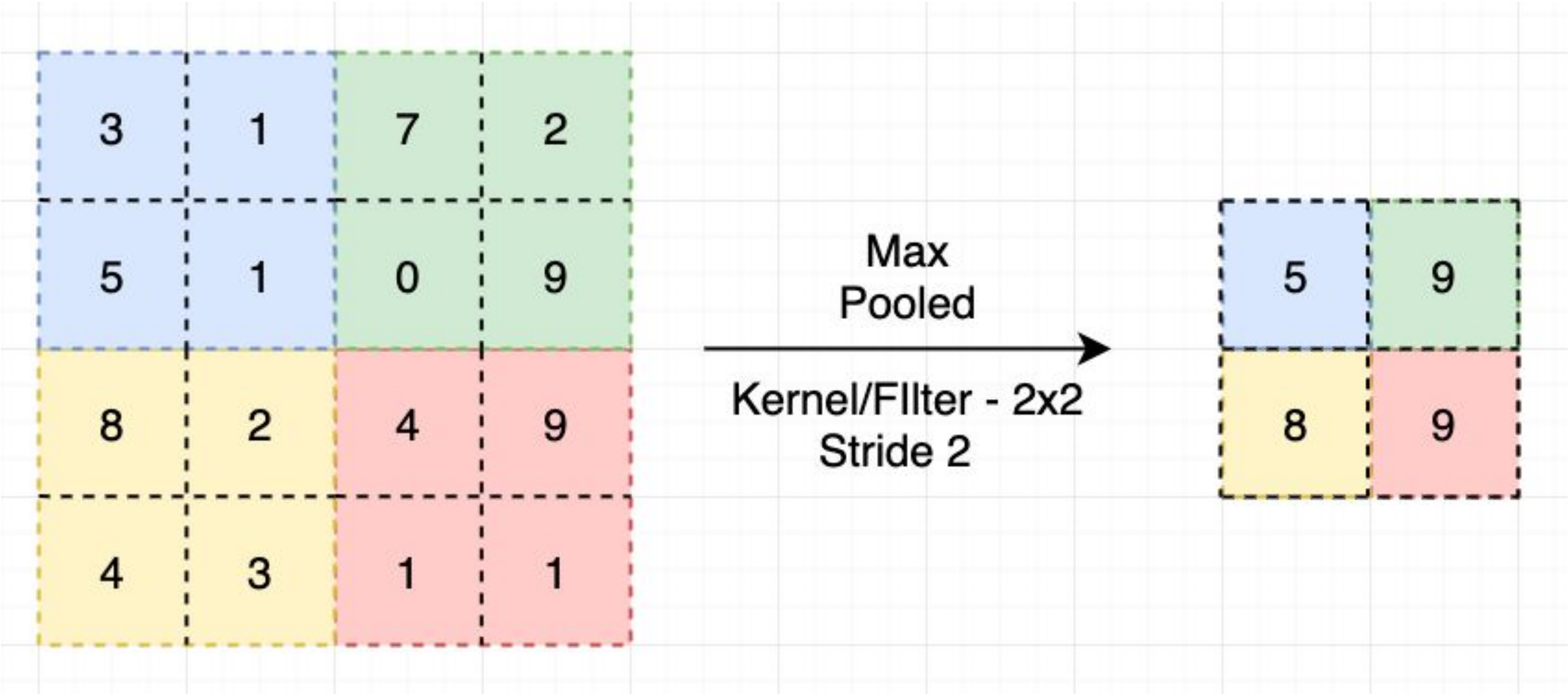
Sigmoid



Linear



Max pooling


3
1
7
2
5
1
0
9
8
2
4
9
4
3
1
1
Max
Pooled
Kernel/FIlter - 2x2
Stride 2
5
9
8
9

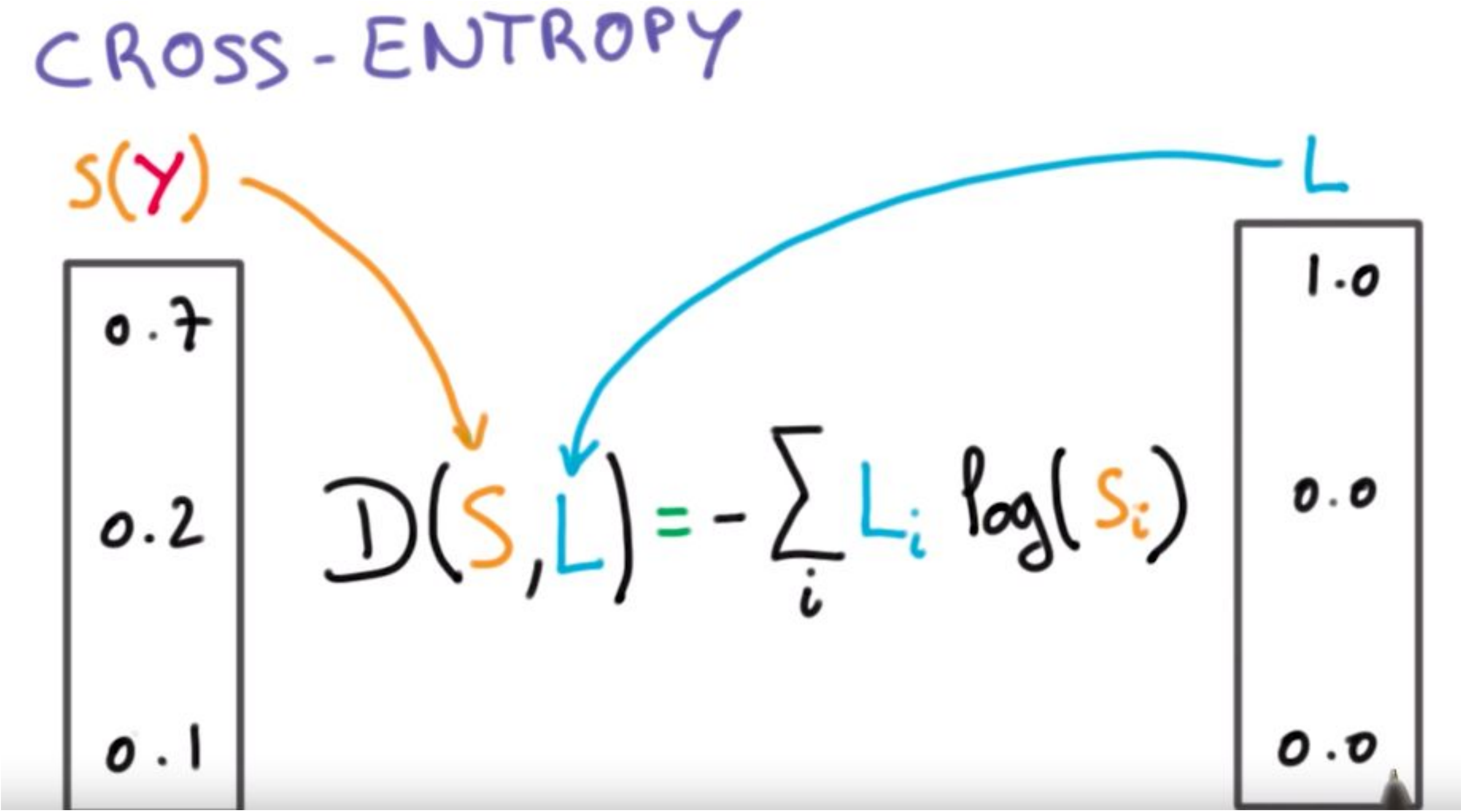
Loss function: Cross-Entropy

True probability distribution
(one-hot)

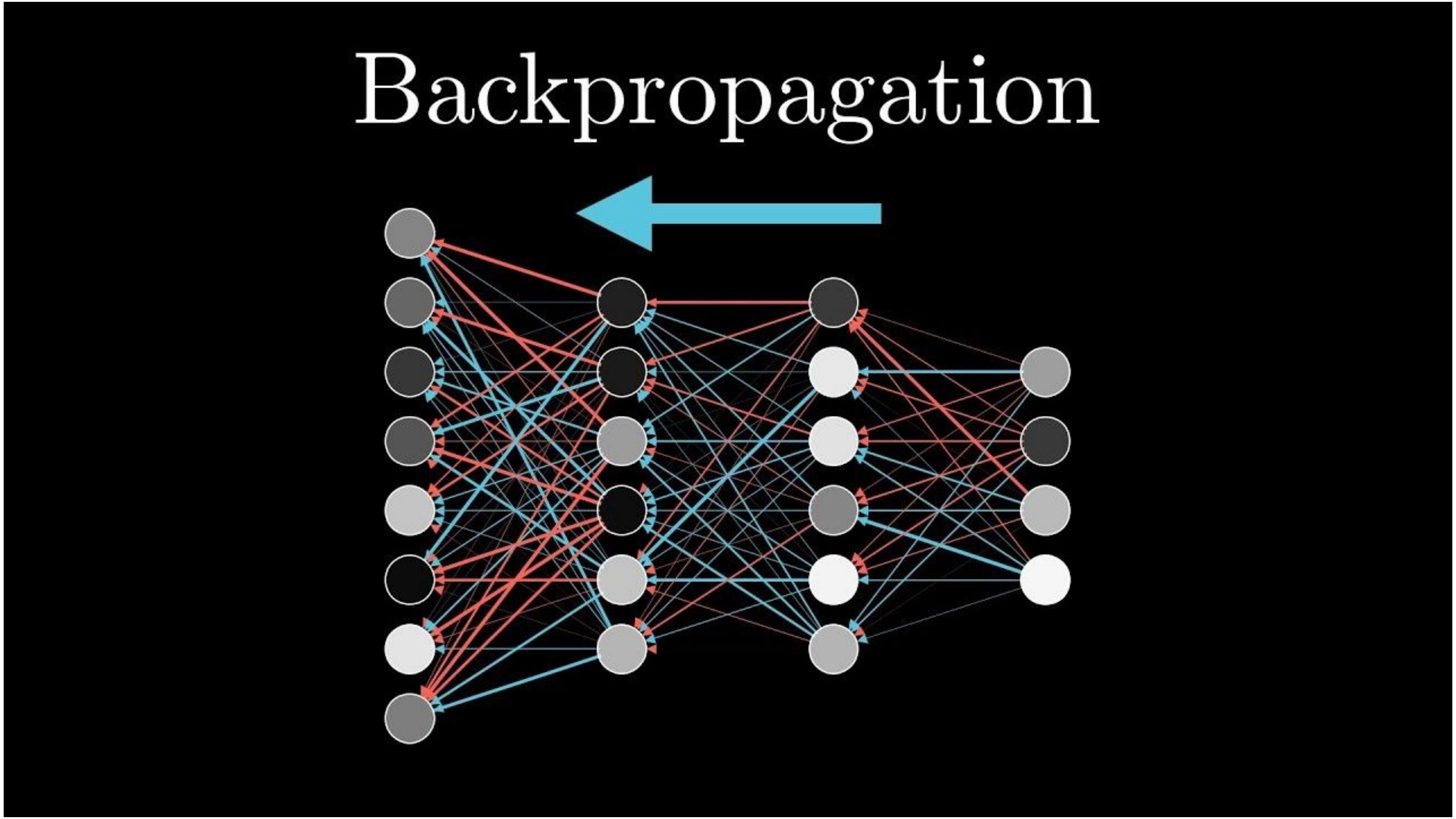
$$H(p,q) = -\sum_{x \in \text{classes}} p(x) \log q(x)$$

Your model's predicted
probability distribution

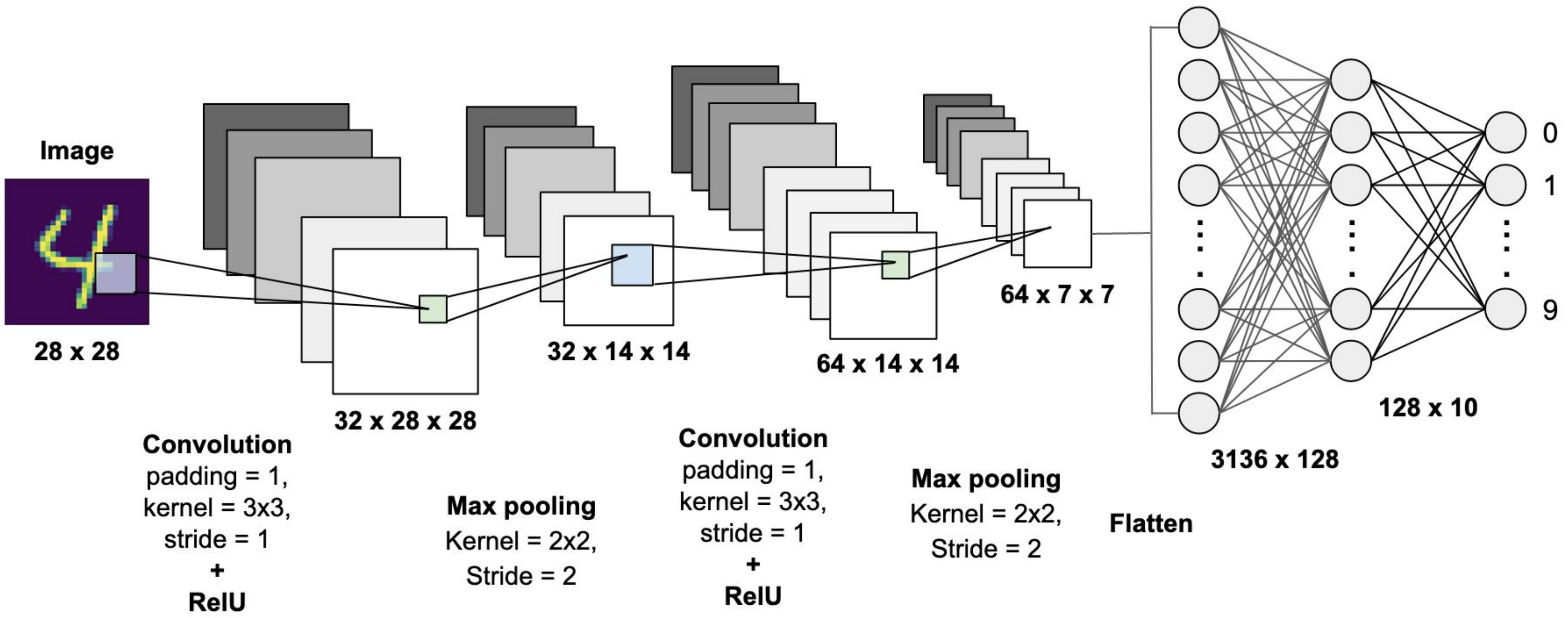
Loss function: Cross-Entropy



Back propagation



Introduction to neural networks with pytorch



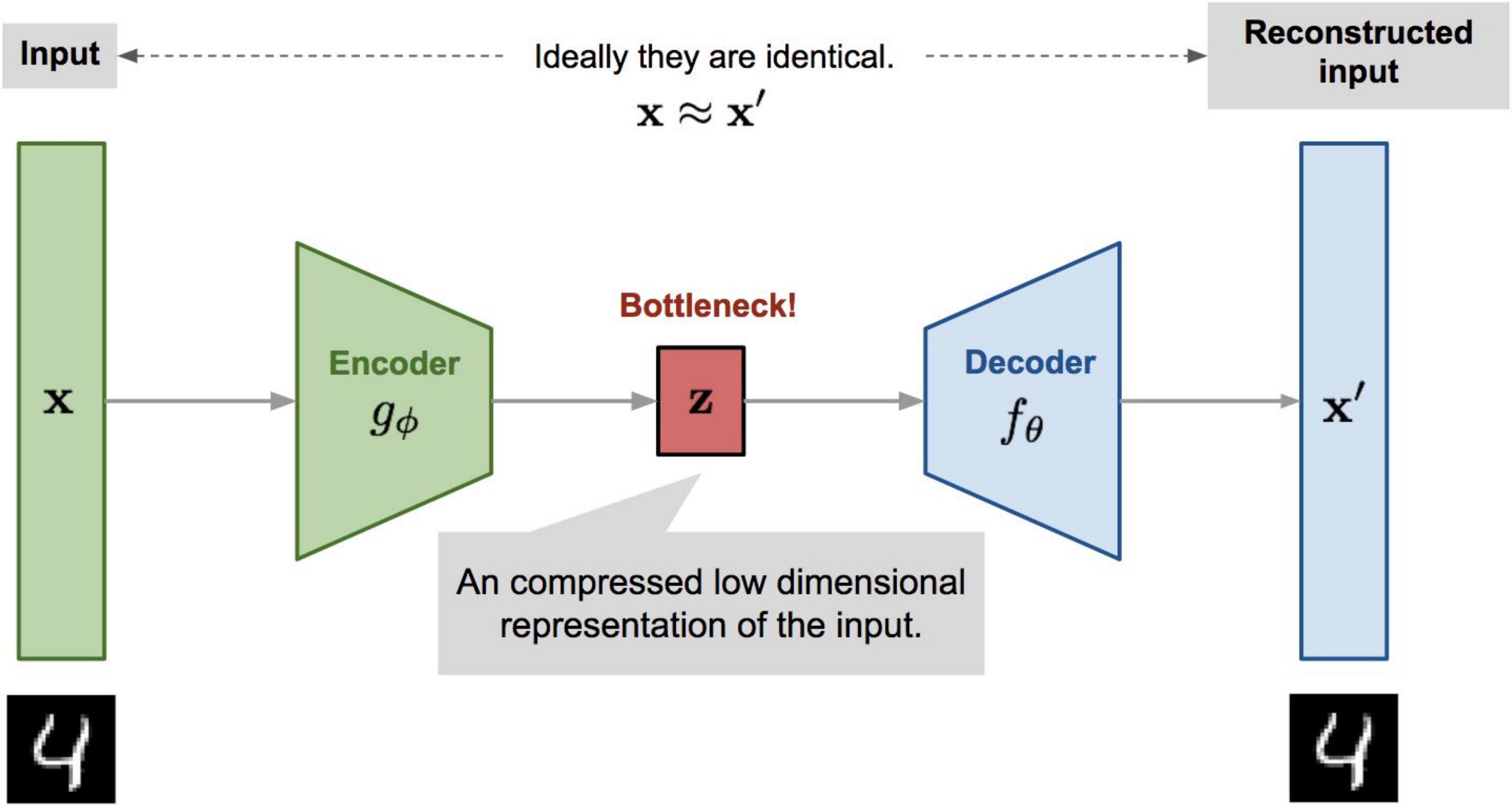
Let’s build a neural network together!

https://bit.ly/3EZ7mMR

Generative models

- Neural networks are not only for classification (supervised learning)!
- Generative models: creating new data with deep neural networks
- example of generative models:
  - Generative adversarial networks (GAN)
  - Variational Autoencoder (VAE)
  - Normalizing Flow

Variational Autoencoder (VAE)



Q&A

Feel free to reach out to me:

joshualin24@gmail.com

Twitter: joshualin24

Github: joshualin24