Deep Learning Software and Pytorch tutorial on Deep Learning 2020

Zhuo Su

2020/10/28

Why am I giving this lecture?

What am I going to talk about?

- Deep Learning Software
- Basic concepts for machine learning
- Coding

Application programming interface (API)

cudnn

gpu

cpu

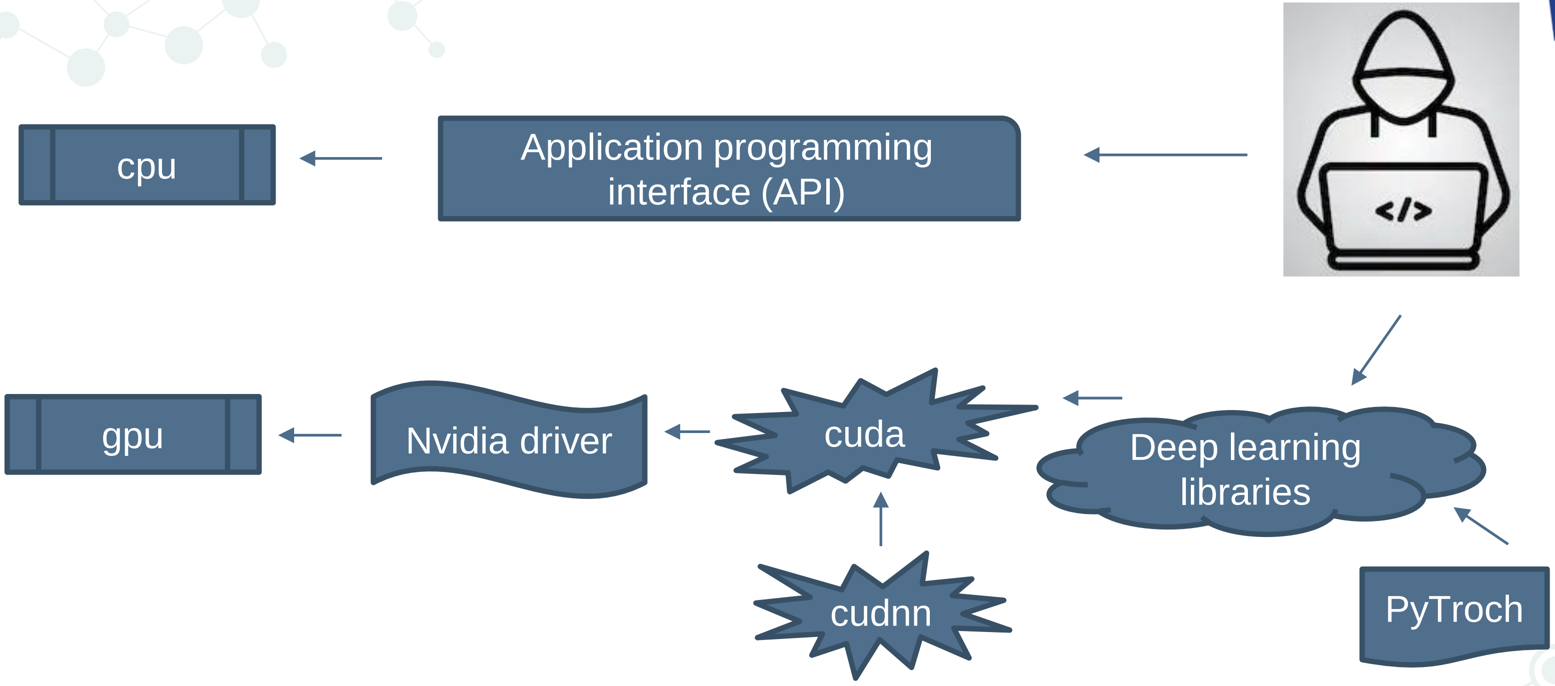
Relationship??

cuda

Nvidia driver

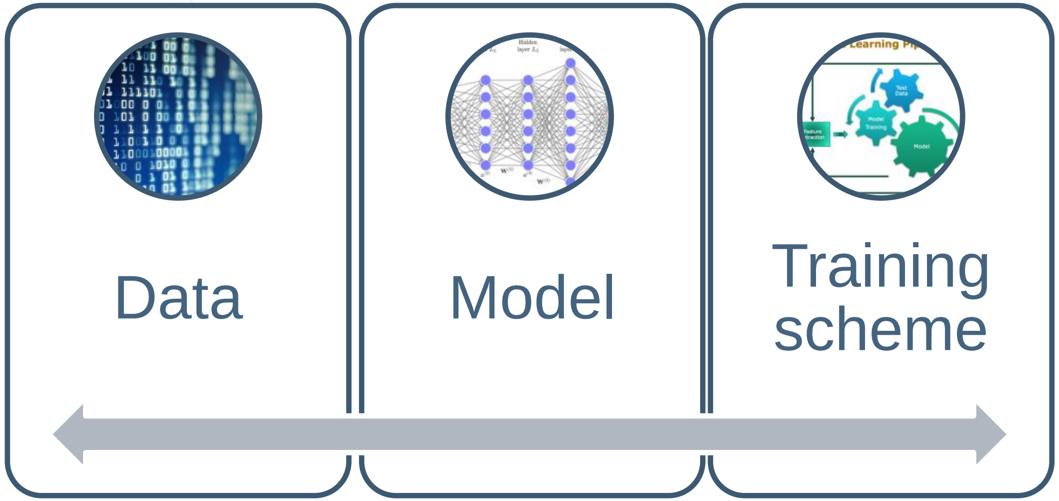
PyTroch

Deep learning libraries



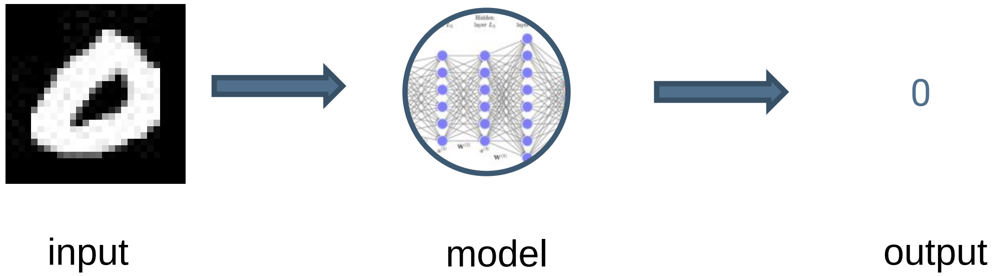
Deep Learning libraries: PyTorch, Tensorflow, Torch, MXNet, Keras, Caffe, etc.

Recipe of a deep learning program

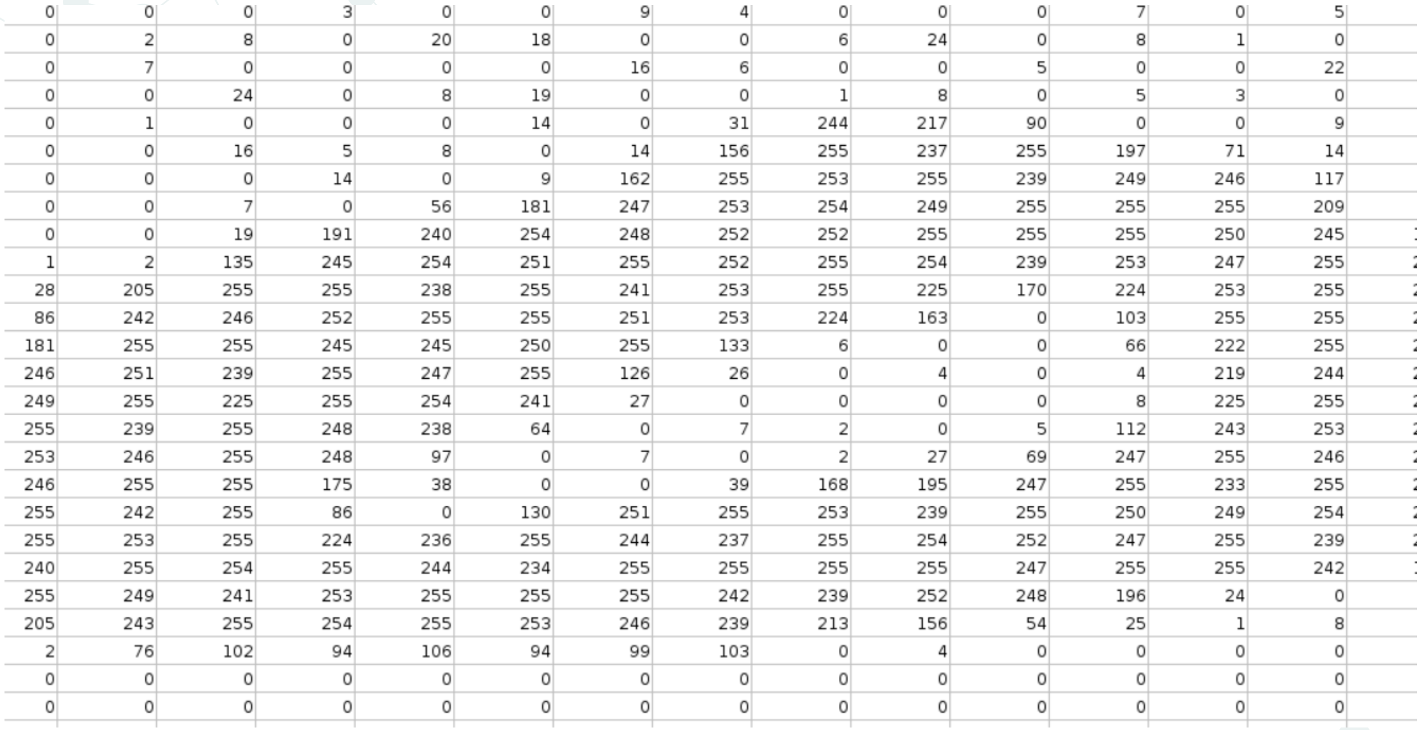


To train a predefined model with a specific scheme, to fit the data

Our targeted problem: Which number did you write?



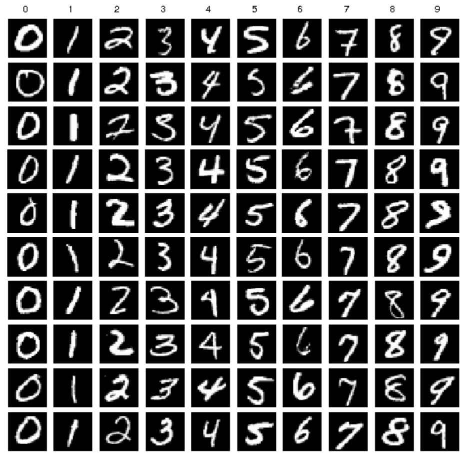
Our targeted problem: Which number did you write?



0	0	0	3	0	0	9	4	0	0	0	7	0	5
0	2	8	0	20	18	0	0	6	24	0	8	1	0
0	7	0	0	0	0	16	6	0	0	5	0	0	22
0	0	24	0	8	19	0	0	1	8	0	5	3	0
0	1	0	0	0	14	0	31	244	217	90	0	0	9
0	0	16	5	8	0	14	156	255	237	255	197	71	14
0	0	0	14	0	9	162	255	253	255	239	249	246	117
0	0	7	0	56	181	247	253	254	249	255	255	255	209
0	0	19	191	240	254	248	252	252	255	255	255	250	245
1	2	135	245	254	251	255	252	255	254	239	253	247	255
28	205	255	255	238	255	241	253	255	225	170	224	253	255
86	242	246	252	255	255	251	253	224	163	0	103	255	255
181	255	255	245	245	250	255	133	6	0	0	66	222	255
246	251	239	255	247	255	126	26	0	4	0	4	219	244
249	255	225	255	254	241	27	0	0	0	0	8	225	255
255	239	255	248	238	64	0	7	2	0	5	112	243	253
253	246	255	248	97	0	7	0	2	27	69	247	255	246
246	255	255	175	38	0	0	39	168	195	247	255	233	255
255	242	255	86	0	130	251	255	253	239	255	250	249	254
255	253	255	224	236	255	244	237	255	254	252	247	255	239
240	255	254	255	244	234	255	255	255	255	247	255	255	242
255	249	241	253	255	255	255	242	239	252	248	196	24	0
205	243	255	254	255	253	246	239	213	156	54	25	1	8
2	76	102	94	106	94	99	103	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	0	0	0	0

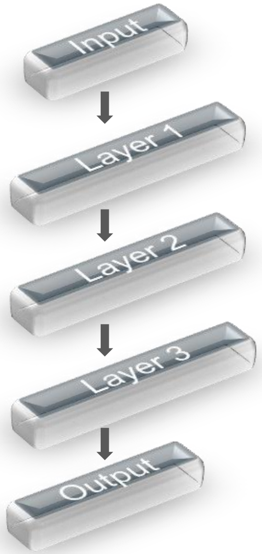
Our targeted problem: Which number did you write?

Data: MNIST database of handwritten digits



http://yann.lecun.com/exdb/mnist/

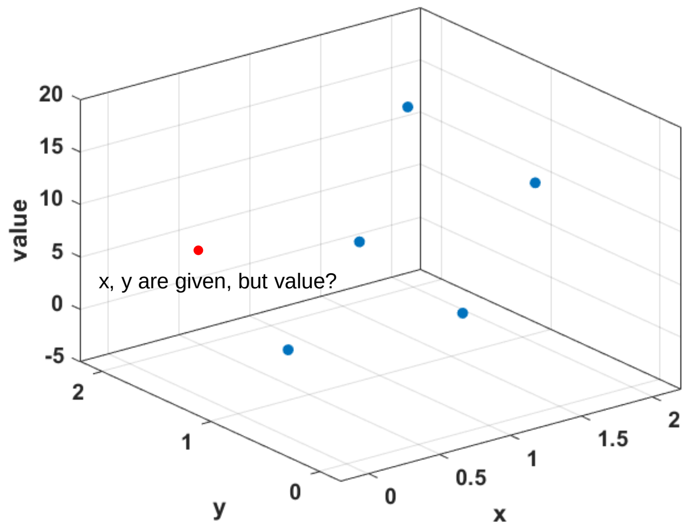
Model (randomly initialized)

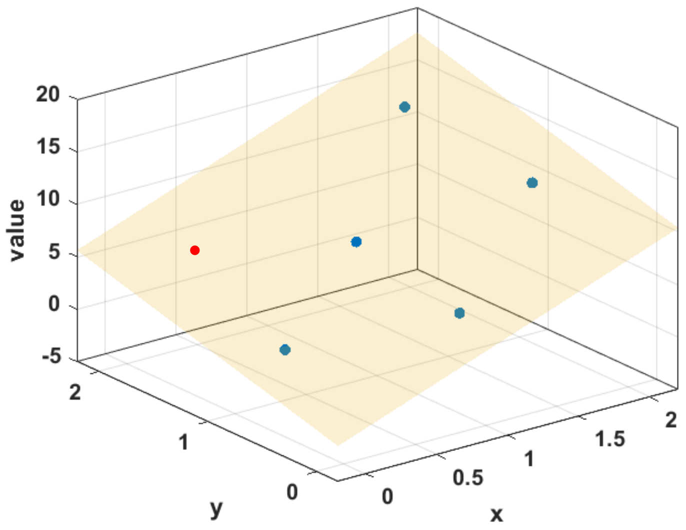


Training scheme

Learning rate?
Optimizer?
Number of iteration?
Loss function?

Before we dig it deeper, let’s see an easier case first…





Data: 5 points

x	y	value
0.8345	0.9785	8.4596
0.0993	0.6754	2.2981
1.8054	1.8001	13.8385
1.8896	0.7385	11.3696
0.9817	0.2224	4.7279

Model: a plane

$f(x, y) = ax + by$

Data: 5 points

x	y	value (ground truth)
0.8345	0.9785	8.4596
0.0993	0.6754	2.2981
1.8054	1.8001	13.8385
1.8896	0.7385	11.3696
0.9817	0.2224	4.7279

Model: a plane

f(x, y) = ax + by

Fitting error:

For point 1: $e_1 = |v_1 - f(x_1, y_1)| = |v_1 - (ax_1 + by_1)|$

For point i: $e_i = |v_i - f(x_i, y_i)| = |v_i - (ax_i + by_i)|$

Total error: $loss = \frac{1}{N}\sum_{i=1}^{N} e_i = \frac{1}{N}\sum_{i=1}^{N} |v_i - (ax_i + by_i)|$

The smaller the loss, the better the model

How can we find the coefficients a and b to minimize the loss?

The solution is simple: using derivatives (a.k.a. gradients in deep learning).

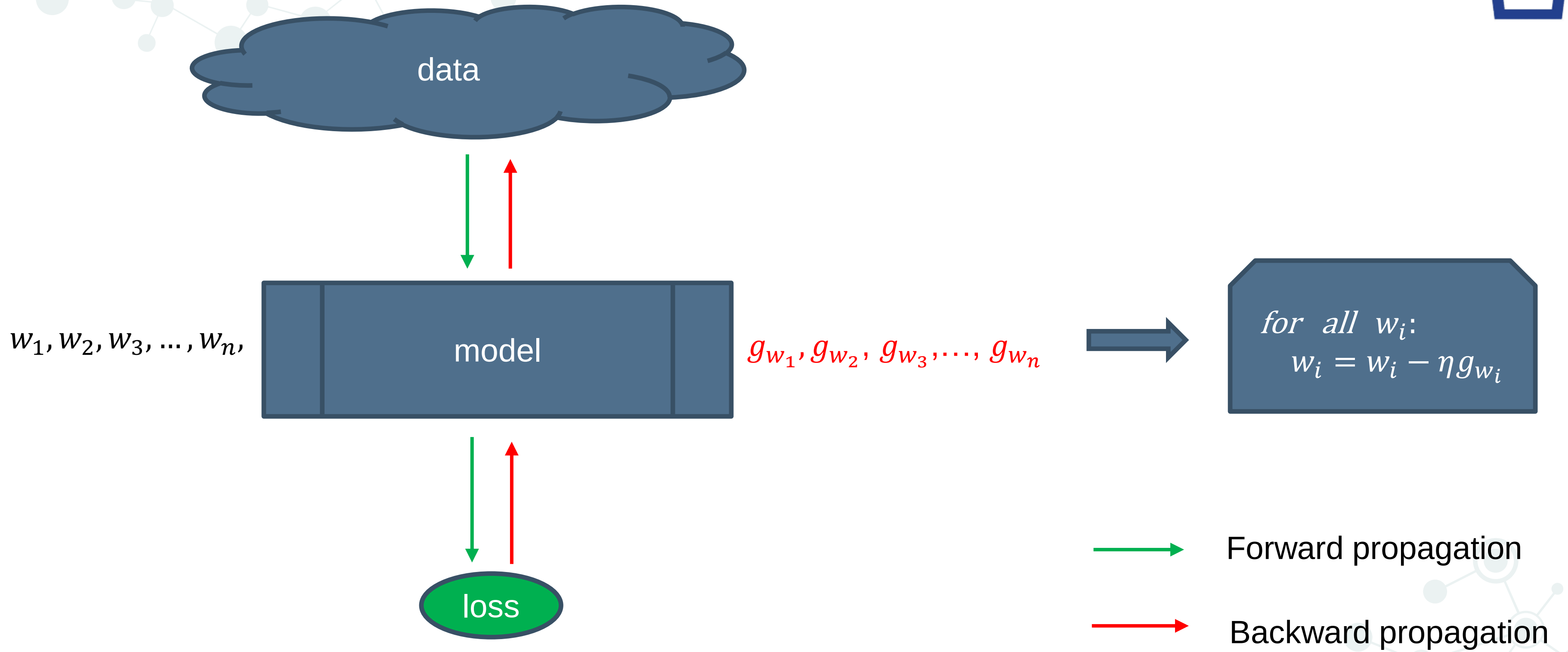
$$g_a = \frac{\partial loss}{\partial a} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial e_i}{\partial a} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial |v_i - (ax_i + by_i)|}{\partial a}$$

$$g_b = \cdots$$

Gradient decent algorithm (updating rule for the coefficients):

```
$Define\ loss,\ learning\ rate\ \eta\ and\ number\ of\ iterations\ n\_iter$
$a = rand()$
$b = rand()$
for $i$ in range(n_$iter$):
    calculate $g_a$ and $g_b$
    $a = a - \eta g_a$
    $b = b - \eta g_b$
return $a$ and $b$
```

Gradient decent algorithm in Pytorch program:


data
$w_1, w_2, w_3, \dots, w_n,$
model
$g_{w_1}, g_{w_2}, g_{w_3}, \dots, g_{w_n}$
loss

$$\begin{aligned}&for \;\; all \;\; w_i:\\ &\quad w_i = w_i - \eta g_{w_i}\end{aligned}$$

Forward propagation

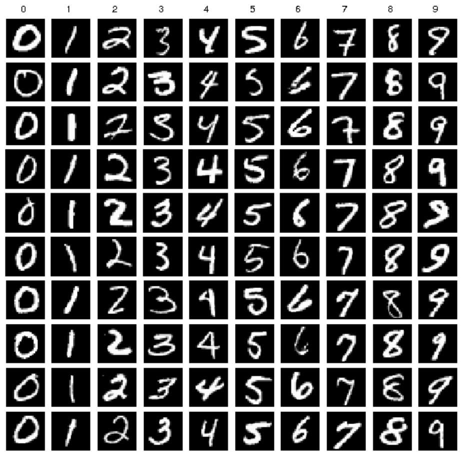
Backward propagation

Tell you something about tensor

And an easy example to code in Pytorch
(please find the script shared in Moodle after the lecture).

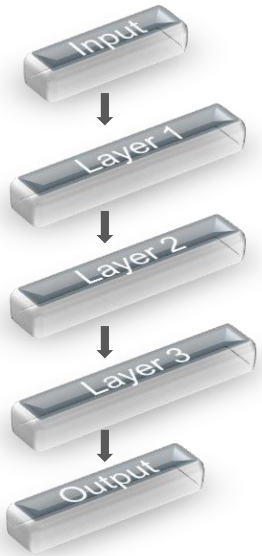
Our targeted problem: Which number did you write?

Data: MNIST database of handwritten digits



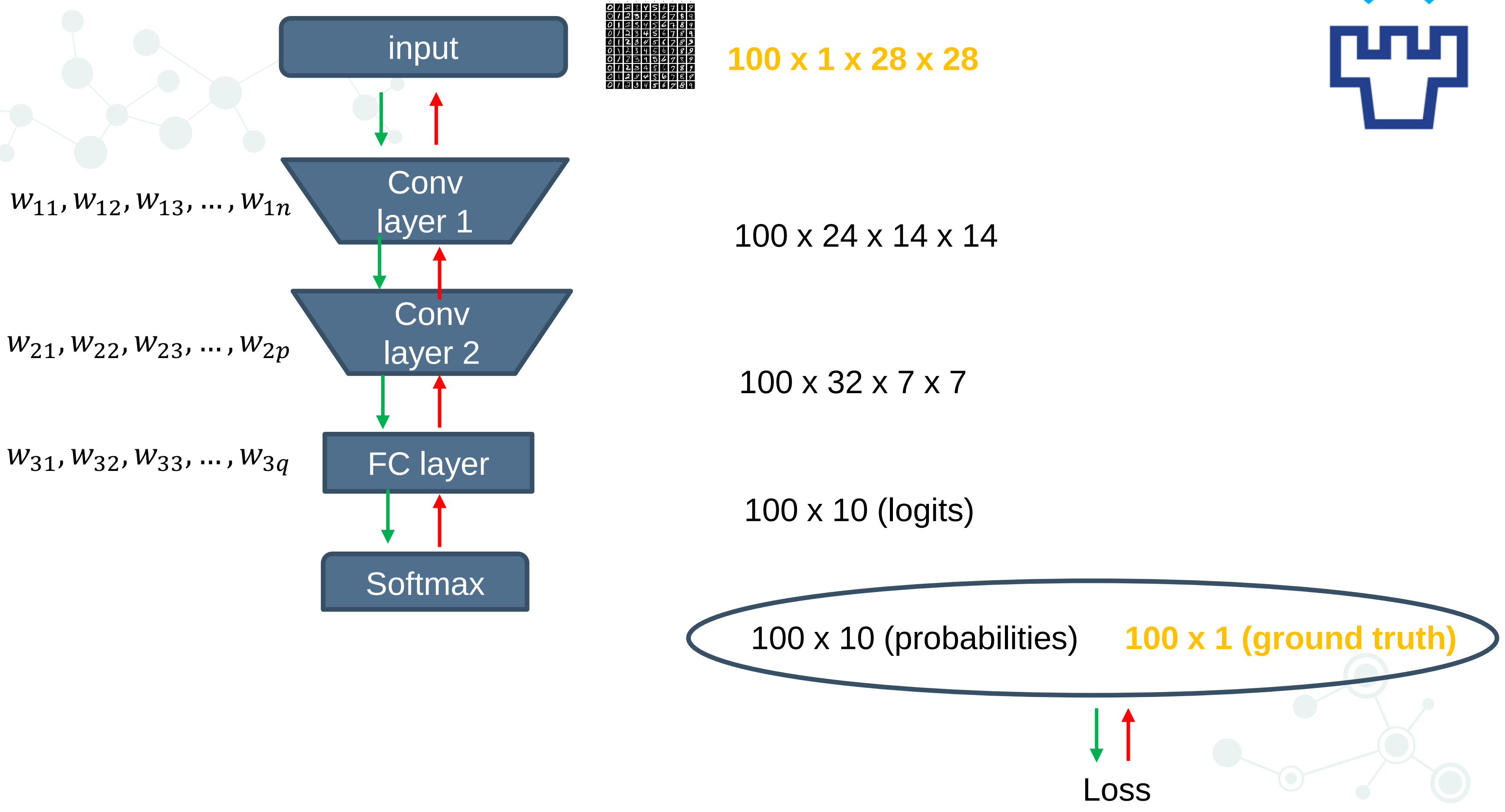
http://yann.lecun.com/exdb/mnist/

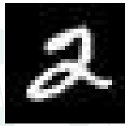
Model
(randomly initialized)



Training scheme

Learning rate?
Optimizer?
Number of iteration?
Loss function?


input
100 x 1 x 28 x 28
$w_{11}, w_{12}, w_{13}, \dots, w_{1n}$
Conv layer 1
100 x 24 x 14 x 14
$w_{21}, w_{22}, w_{23}, \dots, w_{2p}$
Conv layer 2
100 x 32 x 7 x 7
$w_{31}, w_{32}, w_{33}, \dots, w_{3q}$
FC layer
100 x 10 (logits)
Softmax
100 x 10 (probabilities)
100 x 1 (ground truth)
Loss



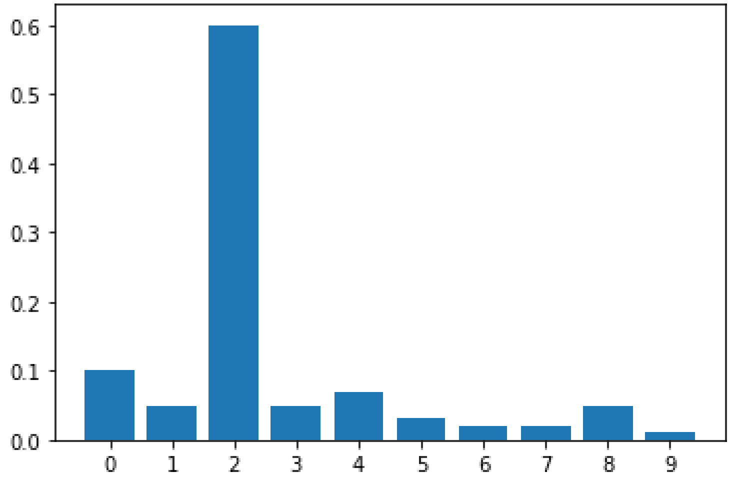
Softmax

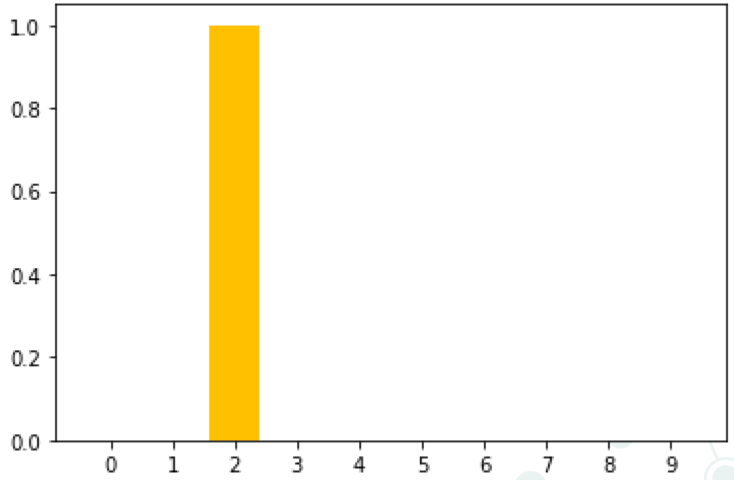
100 x 10 (probabilities)   **100 x 1 (ground truth)**

Loss

0	1	2	3	4	5	6	7	8	9
[0.1,	0.05,	0.6,	0.05,	0.07,	0.03,	0.02,	0.02,	0.05,	0.01]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
[0,	**0,**	**1.0,**	**0,**	**0,**	**0,**	**0,**	**0,**	**0,**	**0]**





Cross entropy (or KL divergence)

Now, lets' code
(please find the script shared in Moodle after the lecture).

Useful links for using Pytorch:

60-minute tutorial:
https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Use Google anytime you come across problems

Attend the lectures, most concepts and fundamentals will be covered

Thanks for listening