Lecture 3: Deep Learning Software and PyTorch tutorial on Deep Learning 2023

Marko Savic, Zhuo Su and Jukka Komulainen

06 November 2023

- Deep Learning Software
- Basic concepts for machine learning
- Coding examples
- Resources
- Assignment 1

Deep learning frameworks

- High-level programming interfaces
  - *Python/NumPy instead of low-level programming with C, C++, CUDA or HIP (CPU + GPU)*
- Provide seamless CPU / GPU usage
  - *Multi-GPU & distributed training of deep neural networks*
- Open source
  - *Very active communities*

Deep learning frameworks

- In fact, tools for defining **static** or **dynamic** general-purpose **computational graphs**
- **Automatic differentiation!**
  - *no need to compute the partial derivatives of millions of parameters by hand*
- All you need to do is to implement forward propagation through a computational graph and specify the cost function and optimizer you want to use!
- The DL frameworks will then compute the derivatives for you, by moving backwards through the graph

Several different options

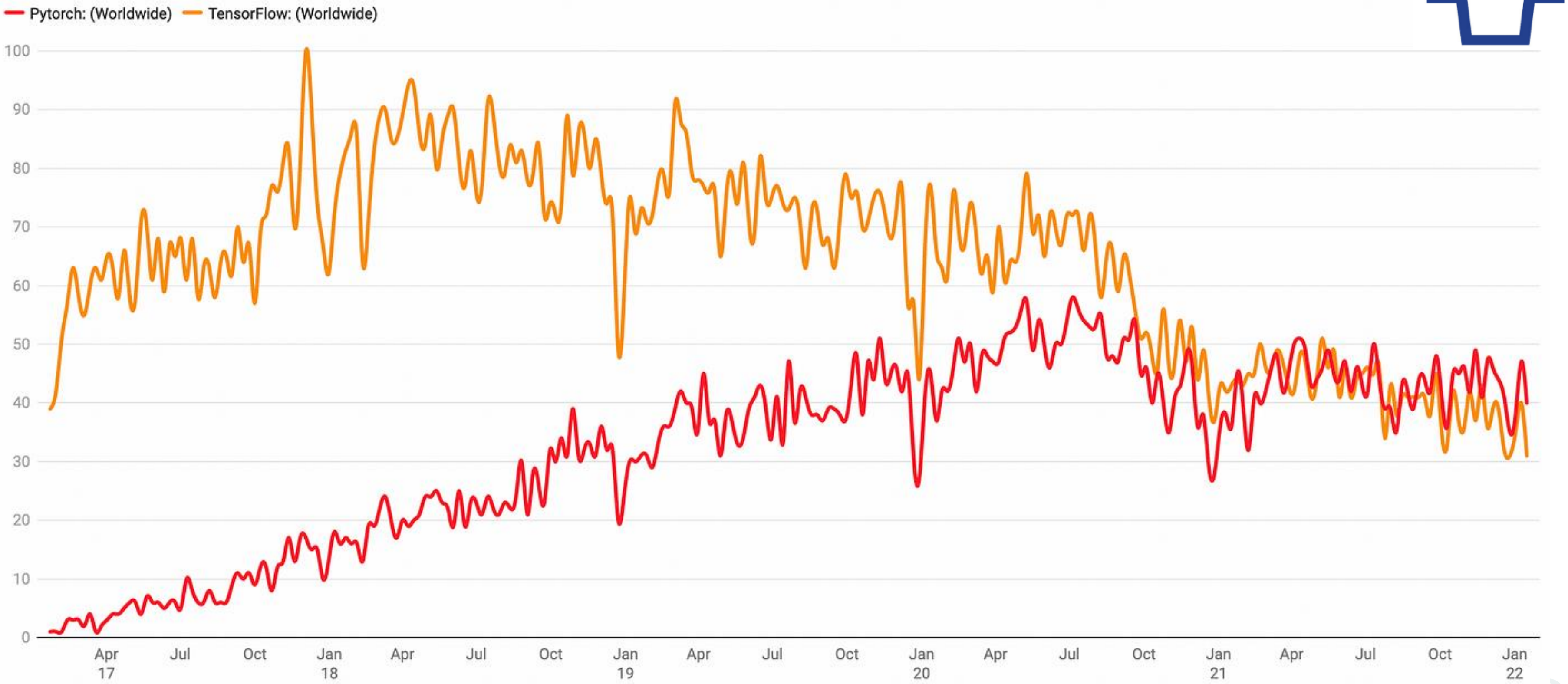
- Different companies and institutes have created their own:
  - *PyTorch*
  - *TensorFlow*
  - *Theano*
  - *Jax*
  - *Caffe*
  - *MXNet*
  - *MATLAB*
  - *…*
- PyTorch and TensorFlow are the most popular ones

PyTorch vs TensorFlow K

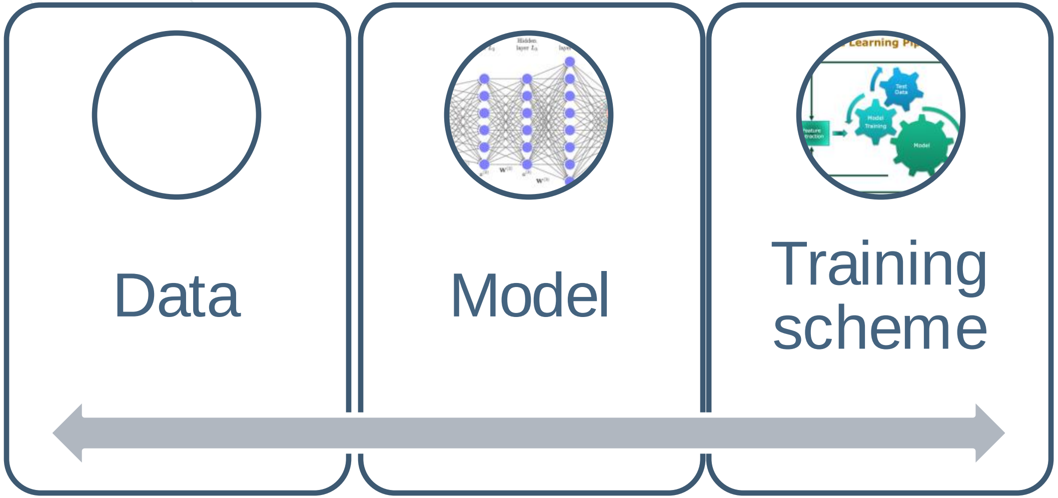
- PyTorch (by Facebook)
  - *Only low-level API: easier than TensorFlow but more low-level than keras*
  - *Easy to debug*
  - *Allows more control and customization, easier experimentation with new architectures*
  - *Great for research use & rapid prototype development*
  - *Catching up with features that are already mature in TensorFlow*

- TensorFlow (by Google)
  - *Low-level API but Keras provides a nice high-level interface*
  - *Difficult to debug*
  - *Keras is easier than PyTorch if you just want to apply deep learning, and not do research in machine learning*
  - *Widely used at production level in industry*
  - *For large-scale deployment*
  - *Lightweight models via TensorFlow Lite*

Google Search Trends PyTorch vs TensorFlow



How to implement a deep learning algorithm



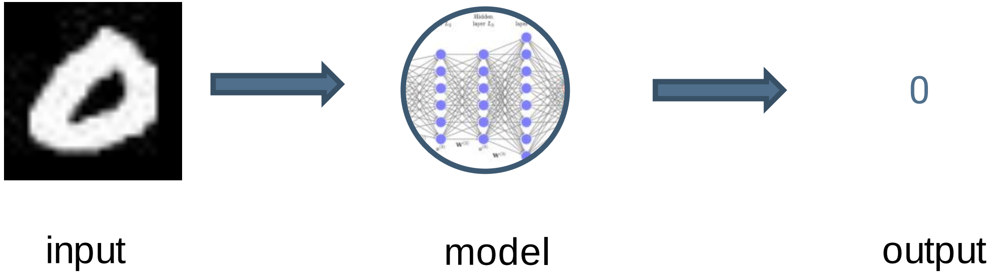
Data

Model

Training scheme

We learn how to solve a problem from the data. Specifically, we train a model (using a training scheme) to fit the data

Problem: Which handwritten digit is in this image?



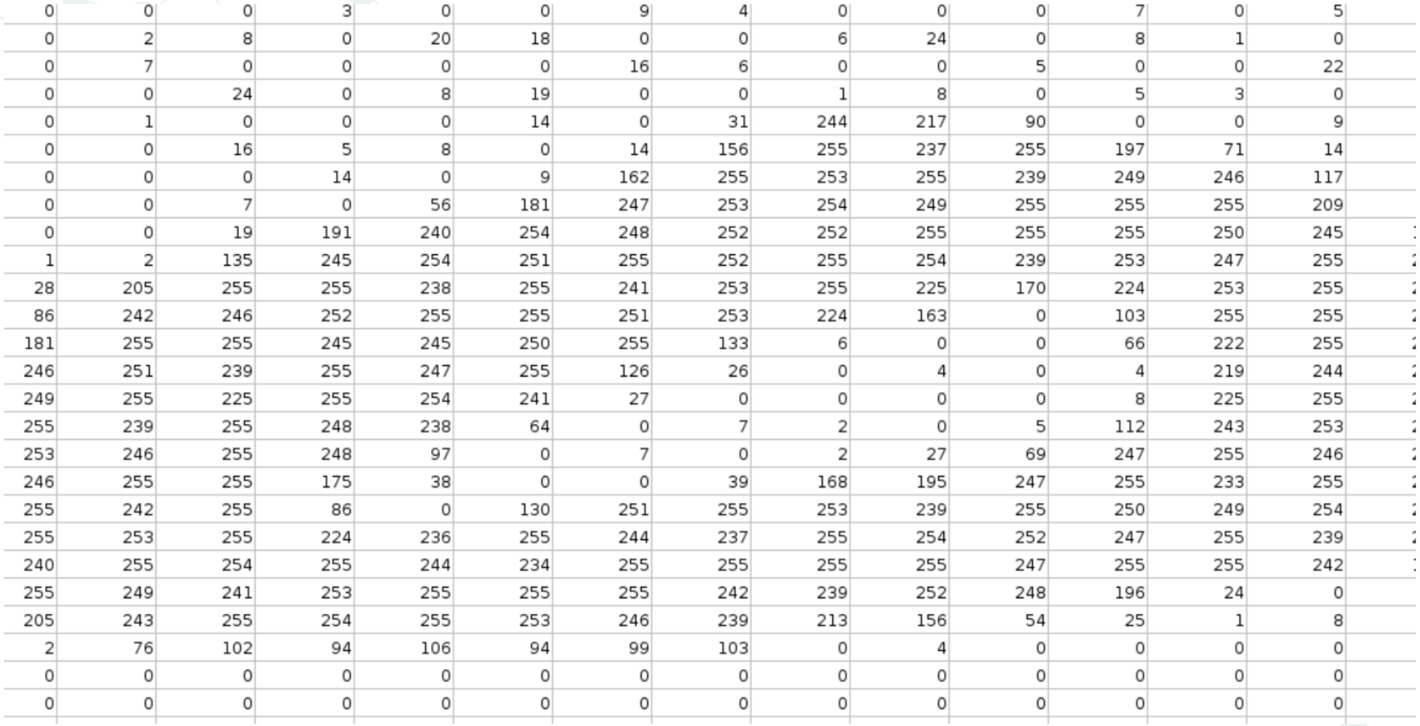
input

model

0

output

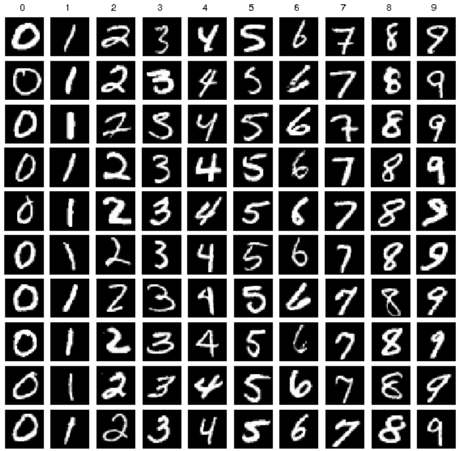
Problem: Which handwritten digit is in this image?



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

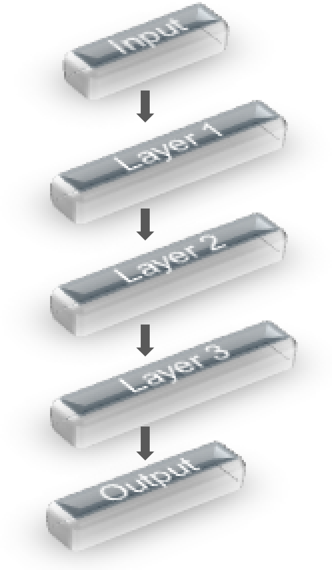
Problem: Which handwritten digit is in this image?

Data: MNIST database of handwritten digits



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

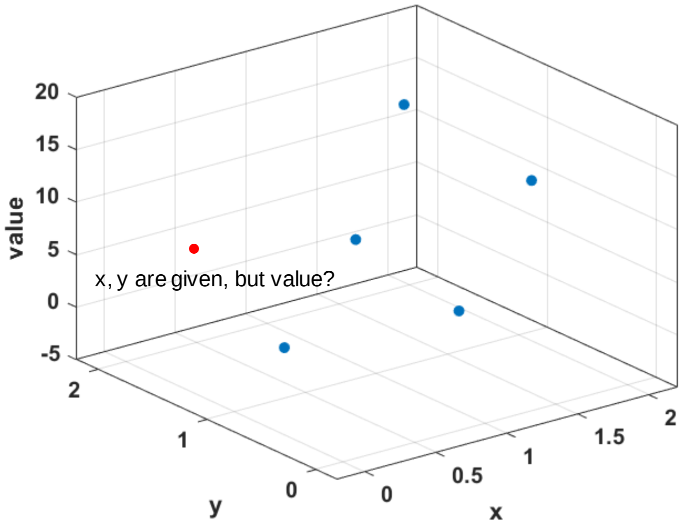
Model
(randomly initialized)

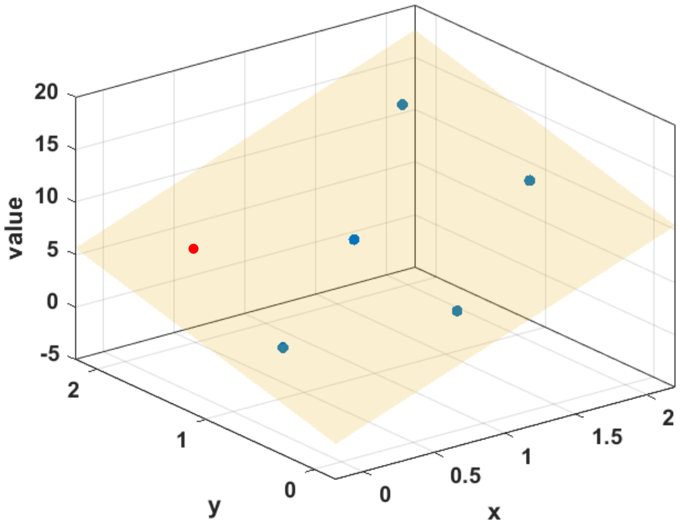


Training scheme

Learning rate?
Optimizer (next lectures)?
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

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

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 + c

Fitting error:

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

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

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

The smaller the cost, the better the model

How can we find the parameters a, b, c to minimize the cost?

The solution is simple: using gradients

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

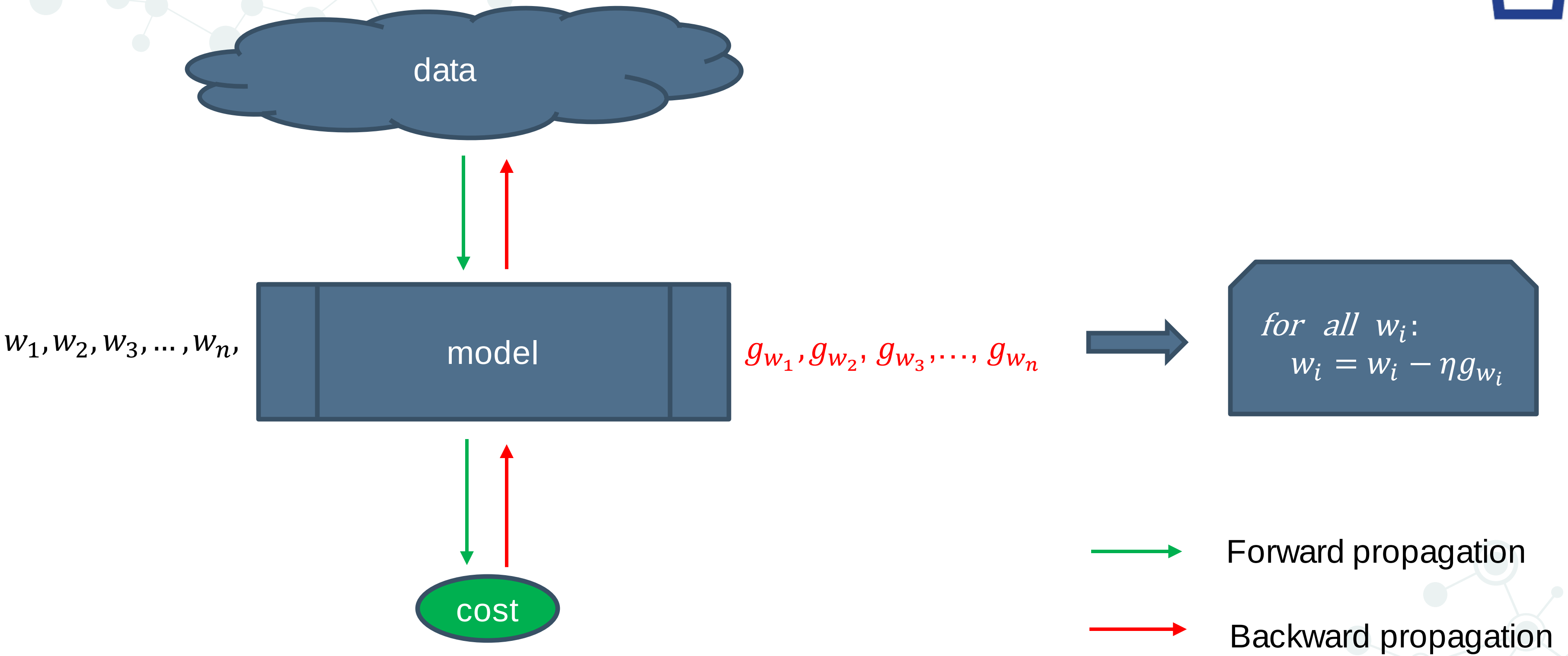
$$g_b = \cdots$$

$$g_c = \cdots$$

Gradient descent algorithm (updating rule for the coefficients):

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

Gradient descent algorithm in PyTorch program:



data

$w_1, w_2, w_3, \ldots, w_n,$

model

$g_{w_1}, g_{w_2}, g_{w_3}, \ldots, g_{w_n}$

$for \;\; all \;\; w_i:$

$w_i = w_i - \eta g_{w_i}$

cost

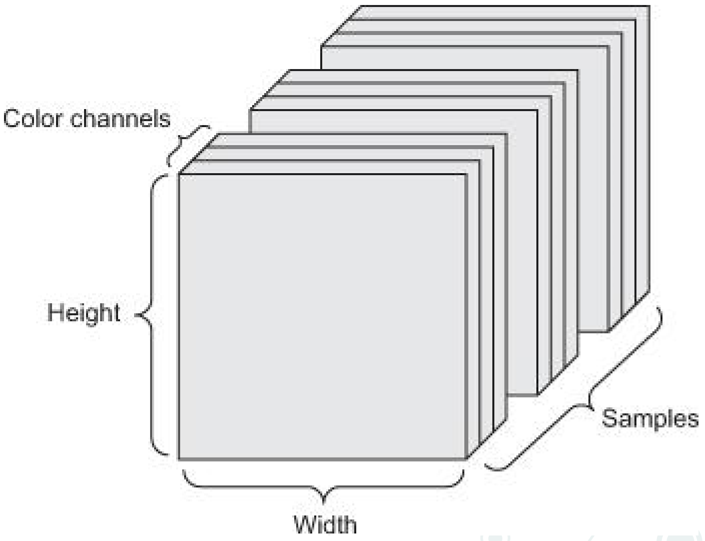
Forward propagation

Backward propagation

What are tensors?

- Generalization of matrices to *n* dimensions
  - *1D tensor: vector*
  - *2D tensor: matrix*
  - *3D, 4D, 5D tensors*

- A set of N, C channel HxW images can be a [N, C, H, W] 4D tensor

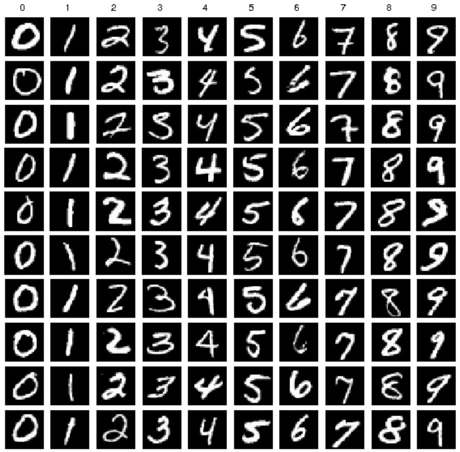


A simple example of code in PyTorch
(please find the script shared in Moodle after the lecture).

When the TA goes on about boring math stuff, but still hasn't taught you how to generate anime waifus/husbandos

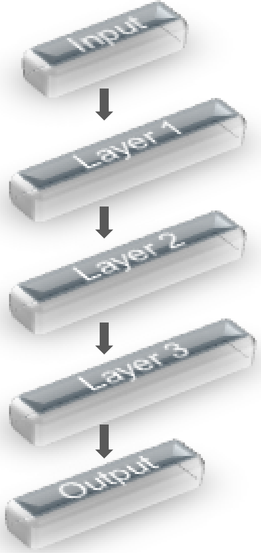
Our targeted problem: Which number did you write?

Data: MNIST database of handwritten digits



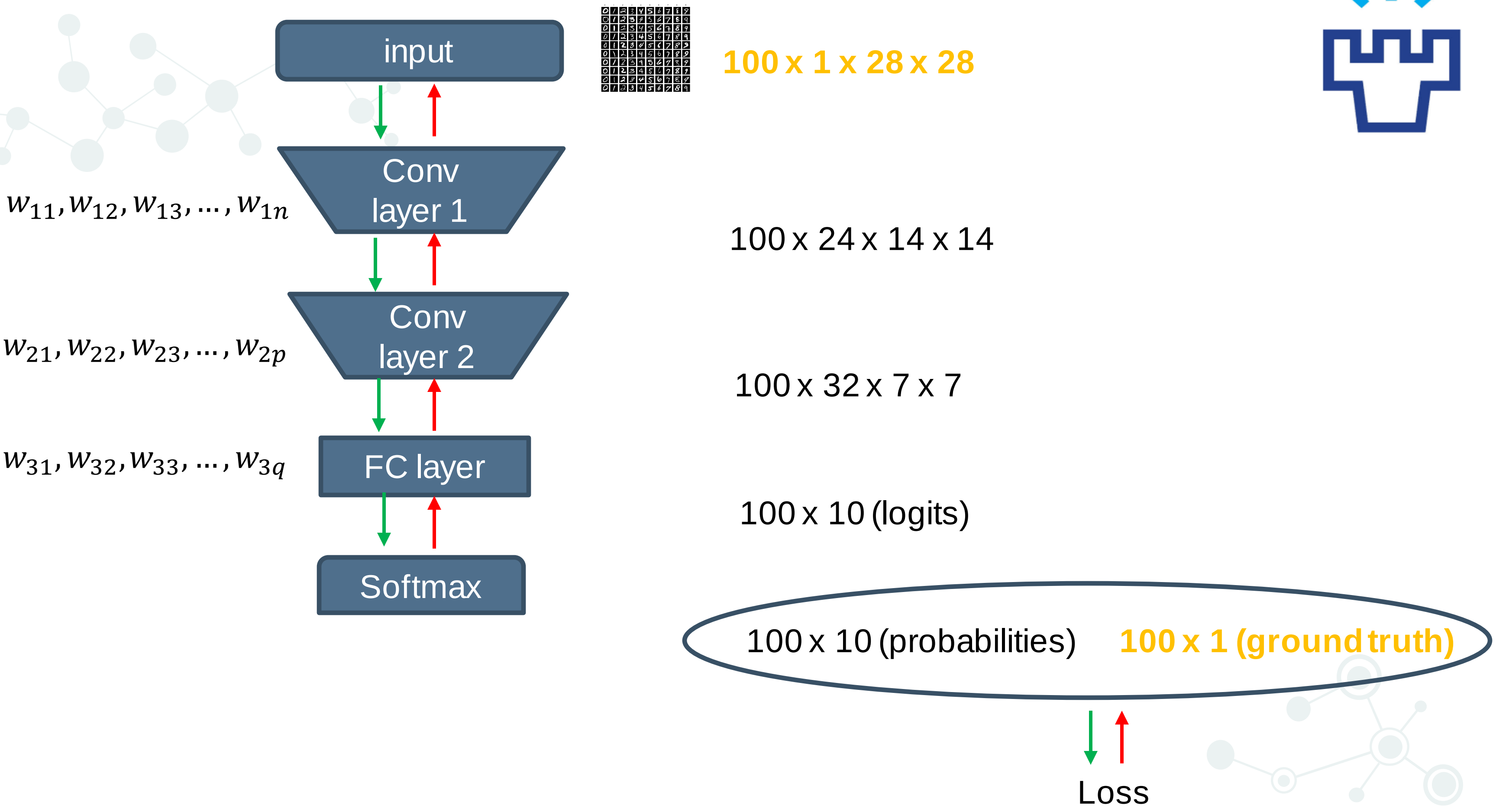
http://yann.lecun.com/exdb/mnist/

Model
(randomly initialized)



Training scheme

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



input

100 x 1 x 28 x 28

$w_{11}, w_{12}, w_{13}, \ldots, w_{1n}$

Conv layer 1

100 x 24 x 14 x 14

$w_{21}, w_{22}, w_{23}, \ldots, w_{2p}$

Conv layer 2

100 x 32 x 7 x 7

$w_{31}, w_{32}, w_{33}, \ldots, w_{3q}$

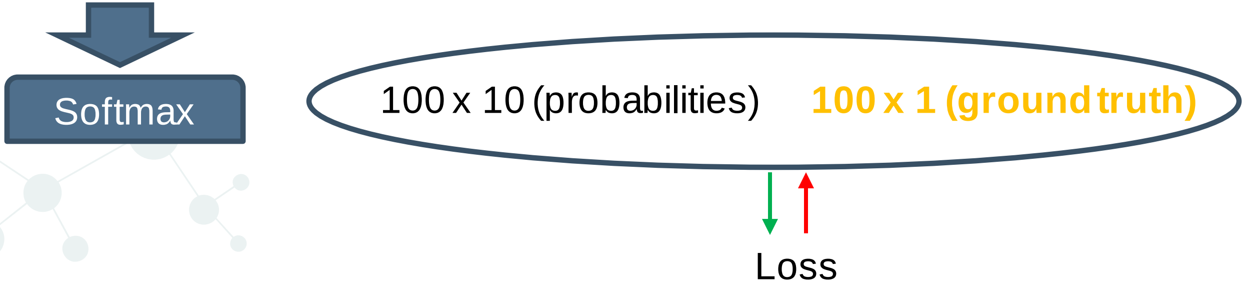
FC layer

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)

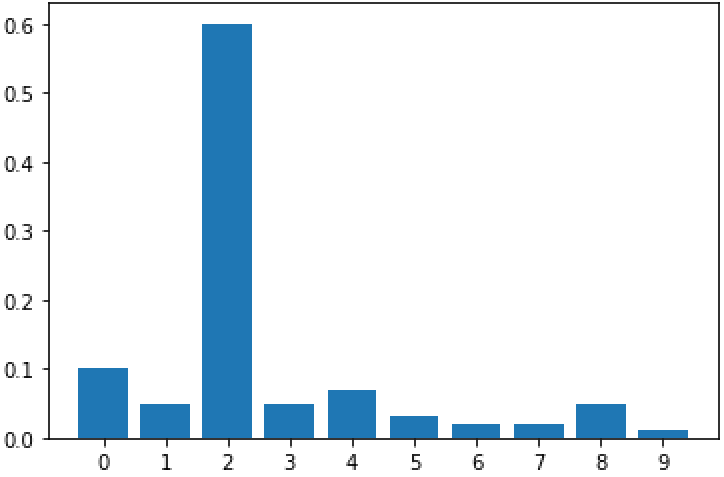
Loss

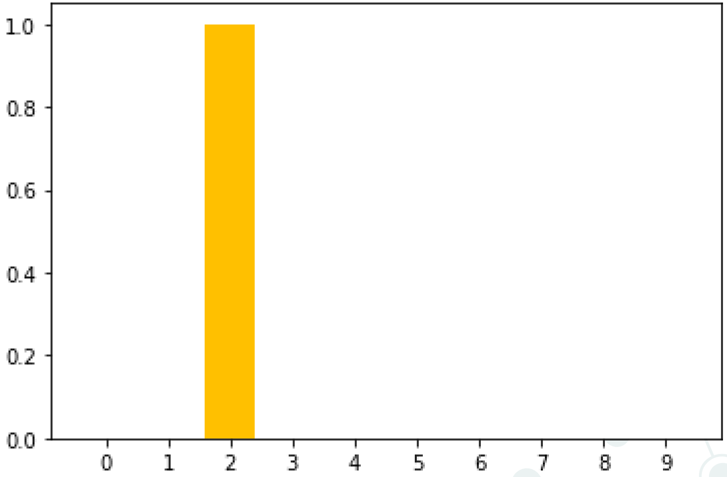
0 1 2 3 4 5 6 7 8 9

[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' take a look at the code
(please find the script shared in Moodle after the lecture).

Useful tips:

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

If you have any questions or need help, write on discord or contact us by email

What resources you can use:

- You own PC (next slide contains details about setting up an environment)
- GPU PCs in TS135 (when vacant, or during our booked slot in December)
- Google Colab with University google account
- CSC notebooks

Environment DL23 and Jupyter

Creating the environment:

Step 1: Go to anaconda website and install miniconda
https://docs.conda.io/projects/miniconda/en/latest/

Step 2: Create isolated conda environment (use supplied .yml file from moodle)

```
conda env create -f environment.yml
source activate DL23
```

OR

In Anaconda Navigator go to Environments -> Import

Using it:

From command line (or also simply from UI if you have windows):

```
conda activate DL23
```

After that run jupyter notebook from the activated DL23 environment:

```
jupyter-notebook
```

OR

```
jupyter-lab
```

Cloud resources

Google Colab:

Step 1: If you haven't already, follow patio instructions on how to get university google accout:
https://patio.oulu.fi/en/services-and-instructions/it-services/information-systems/google-workspace-education

Step 2: Go to:
https://colab.research.google.com

Step 3: Upload the notebook you want to work on.

CSC Notebooks:

Step 1: Go to and login:
https://notebooks.rahtiapp.fi/

Step 2: Select "Practical Machine learning" and run

Step 3: Upload the notebook you want to work on.

Let's take a quick look at assignment 1

Thanks for listening