

Sztuczne sieci neuronowe

laboratorium nr 1

Agenda

1. Wstęp
2. BHP
3. Tematyka zajęć
4. Zasady zaliczenia




Informacje organizacyjne

- stacjonarnie + Teams
 - ◆ **1dm0giz** - utworzyć folder **IMIE_NAZWISKO_NR-ALBUMU**
- co poniedziałek (jeśli wypadną -> @)
- starosta?
- godzina rozpoczęcia? (15 min **później** start, **przerwa** czy 15 min **wcześniej** koniec)



Na zajęciach:

- **BHP** - podpisy na liście
 - punktualność (**kwadrans** akademicki)
 - obecność obowiązkowa (max. **3** nieobecności) - lista na każdych zajęciach
 - **usprawiedliwienie** nieobecności od lekarza lub PBS
 - telefony, przerwy, wyjścia...
 - picie i jedzenie
 - komputery własne lub lab.
- 

BHP



Plan laboratorium

Lab. 1 → powtórka z NumPy, wprowadzenie do PyTorch


Lab. 2 i kolejne → +/- zgodnie z wykładem

Tematyka: **sztuczne sieci neuronowe**

Adekwatne oprogramowanie:

- środowisko programistyczne **Python 3.X** (**Visual Studio Code, Jupyter Notebook, Google Colab, pyCharm, Tonny, Notepad**) <https://code.visualstudio.com/docs/languages/python>, <https://colab.research.google.com/> (lokalnie)
- środowisko uczelniane - <https://jupyter.io.pbs.edu.pl>

Biblioteki: NumPy, **PyTorch**, być może także **Keras / TensorFlow**



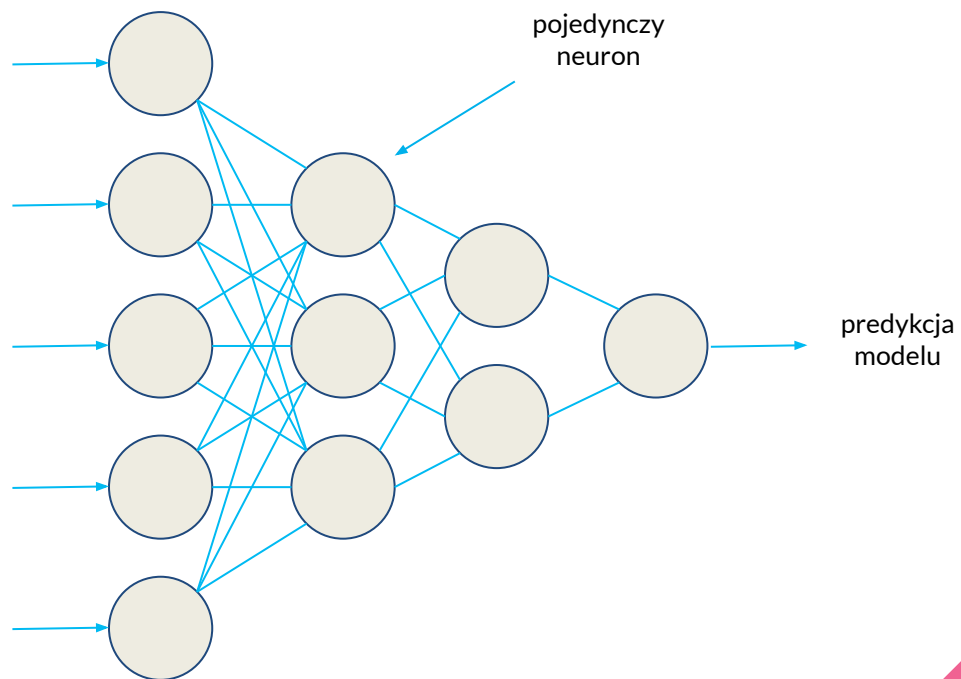
Zasady zaliczenia

1. Zadania **samodzielnie** wykonane na zajęciach (chat gpt **X**)
2. Dwa **kolokwia** - w połowie i pod koniec semestru
3. Dodatkowe plusy za **pyData / aktywność na zajęciach**
4. **Notebooki** wykonane na zajęciach na **Teams**
5. **>3** nieobecności nieusprawiedliwione → **n.p.**

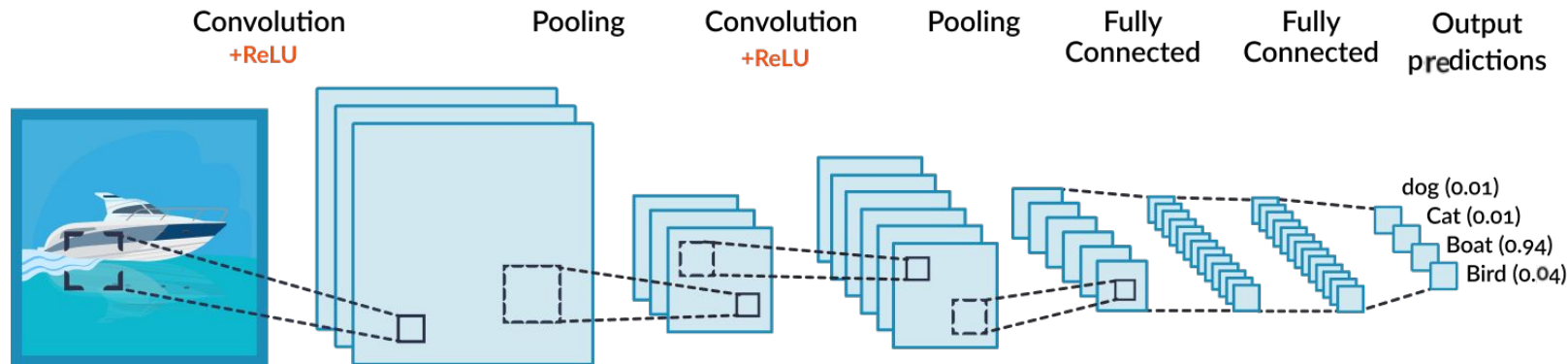
Jeśli jest problem ze zrozumieniem zadania /
techniczny / zdrowotny / jakikolwiek inny → **zgłaszać od razu!**



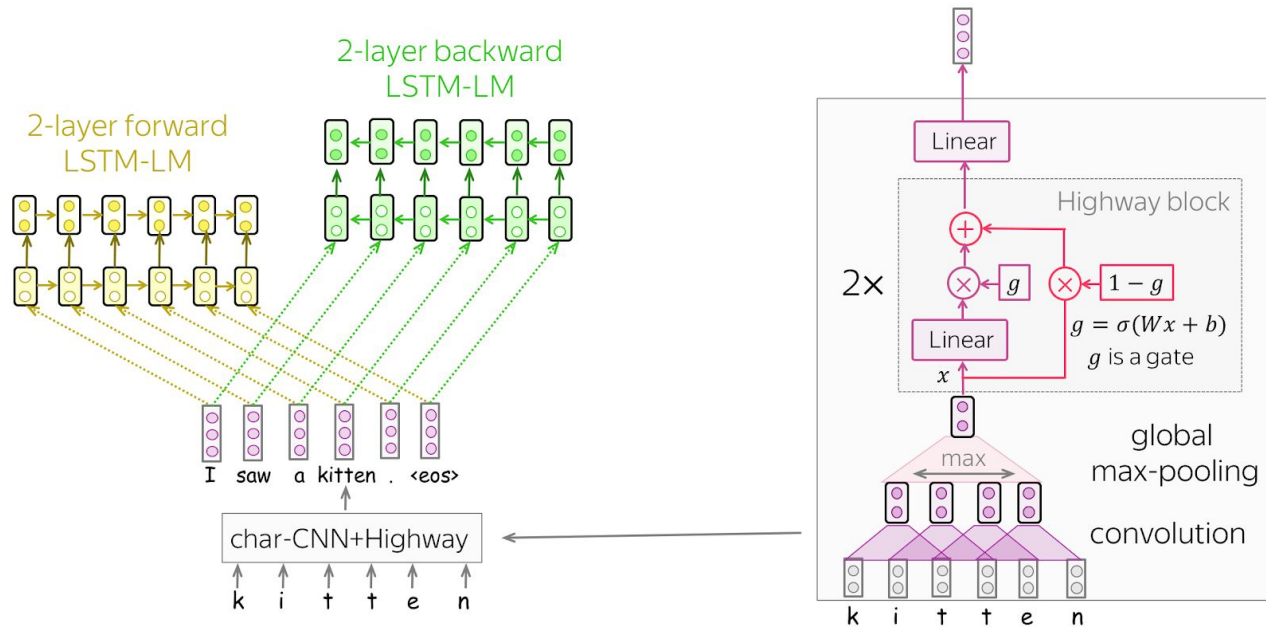
Sieci neuronowe



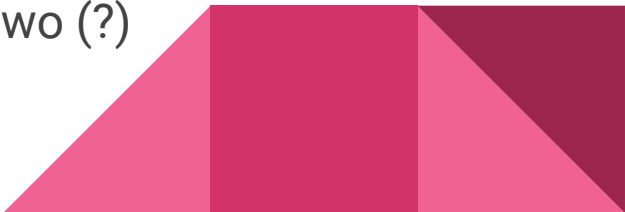
Sieci neuronowe - widzenie maszynowe



Sieci neuronowe - przetwarzanie języka naturalnego



Matematyka w sieciach neuronowych

- wektor, macierz, tensor
 - dodawanie, mnożenie, iloczyn skalarny, transpozycja, ...
 - pochodna (cząstkowa) funkcji wielu zmiennych, gradient
 - logarytm, e^x , $|x|$ i inne podstawowe funkcje matematyczne
 - średnia, odchylenie standardowe, prawdopodobieństwo (?)
- 

Matematyczne podstawy sztucznej inteligencji - co było?

Ćwiczenia laboratoryjne:

1. NumPy: listy pythonowe kontra wektory numpy, porównanie czasu działania, praca na tensorach, podstawowe funkcje
2. SciPy: podstawowe funkcje
3. PyTorch: pakiet autograd
4. Metody numeryczne, minimalizowanie funkcji ciągłej: algorytm najszybszego spadku



Sieci neuronowe w Pythonie

PYTORCH

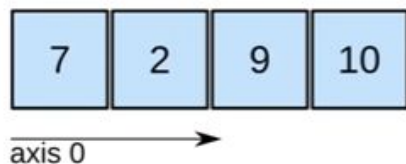
 Keras

 TensorFlow™

inne: **JAX** (nowość), Caffe, Theano, CNTK, ...

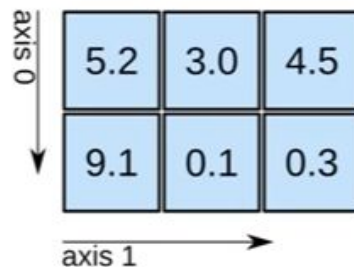
NumPy array

1D array



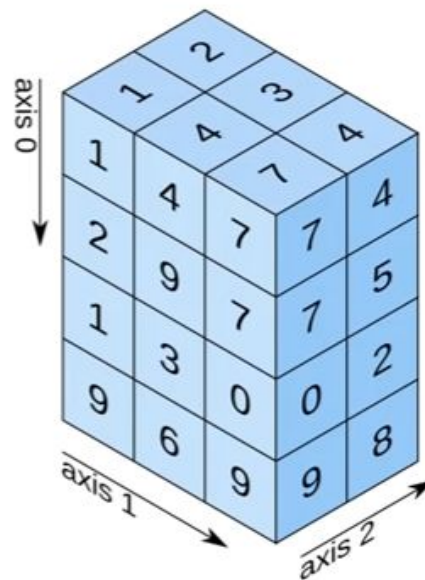
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)

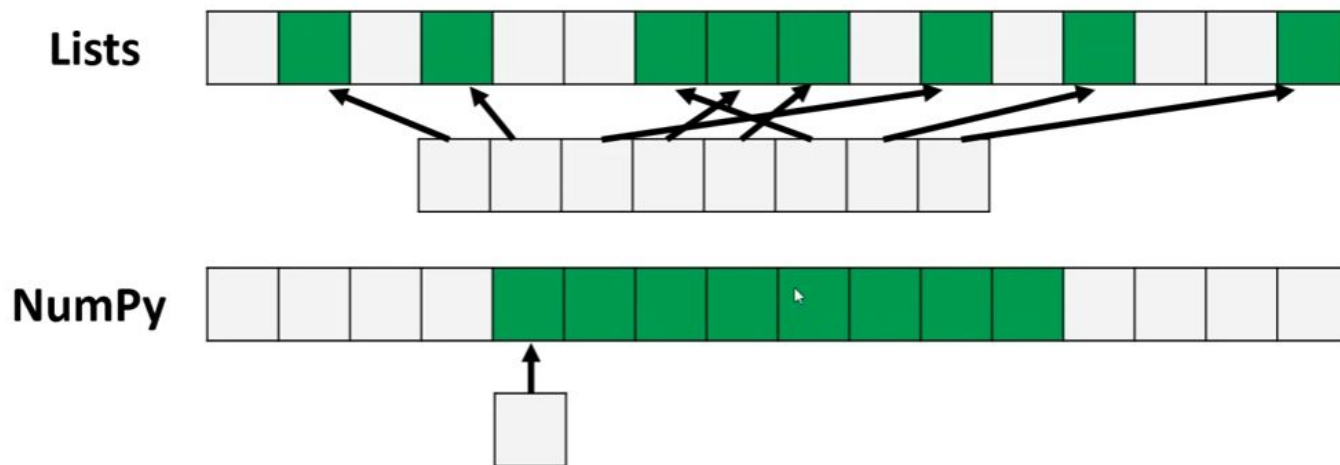
NumPy array

- szybsze niż listy, bo:
 - ustalony typ elementów (np. int32, float64)
 - potrzeba mniej informacji o elementach → mniej pamięci
 - typ elementów nie jest sprawdzany podczas iteracji
- elementy zajmują sąsiednie bloki w pamięci
 - tzw. operacje SIMD - Single Instruction Multiple Data (wektoryzacja)



NumPy array

Why is NumPy Faster? - Contiguous Memory



NumPy array

Computations with Arrays

- Rule 1:** Operations between multiple array objects are first checked for proper shape match.
- Rule 2:** Mathematical operators (+ - * / exp, log, ...) apply element by element, on the values.
- Rule 3:** Reduction operations (mean, std, skew, kurt, sum, prod, ...) apply to the whole array, unless an axis is specified.
- Rule 4:** Missing values propagate unless explicitly ignored (nanmean, nansum, ...).

Polecane materiały - NumPy

Introduction to Numerical Computing with Numpy

<https://github.com/enthought/Numpy-Tutorial-SciPyConf-2020/blob/master/slides.pdf>

A Visual Intro to NumPy and Data Representation

<http://jalammar.github.io/visual-numpy>

Patryk Miziula, *Wektoryzowanie danych w Pythonie - szybciej i matematyczniej*

<https://www.youtube.com/watch?v=8ySi14GI7hU>



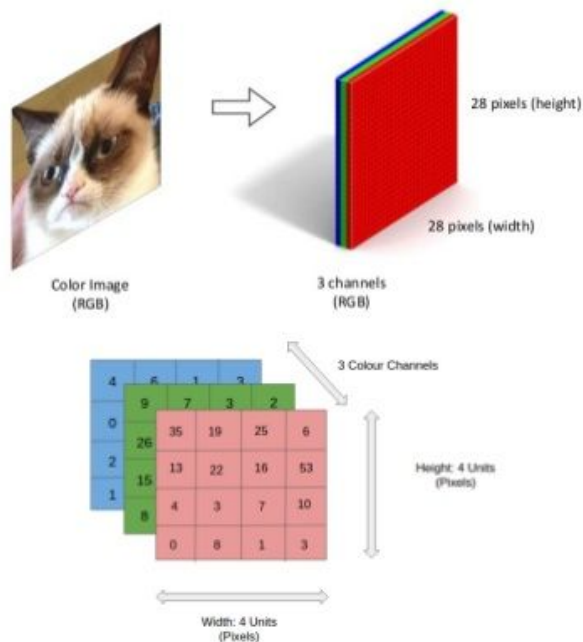
PyTorch

“NumPy na sterydach”

- `numpy.array` → **`torch.Tensor`**
- wsparcie dla obliczeń na GPU (CUDA)
- *autograd* - silnik do automatycznego obliczania pochodnych funkcji
- implementacje poszczególnych warstw i całych modeli sieci neuronowych, mechanizmów ładowania danych, algorytmów optymalizacji funkcji straty itd.

Obrazek RGB jako tensor

color image is 3rd-order tensor



Tensory 4-wymiarowe

→ 0

(4,)



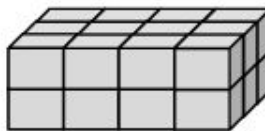
0 → 1

(2, 4)



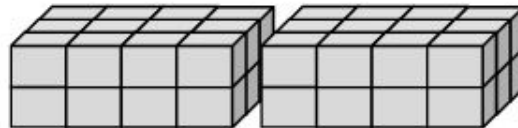
0
1 → 2

(3, 2, 4)



0
1
2 → 3

(2, 3, 2, 4)



Tensory N-wymiarowe

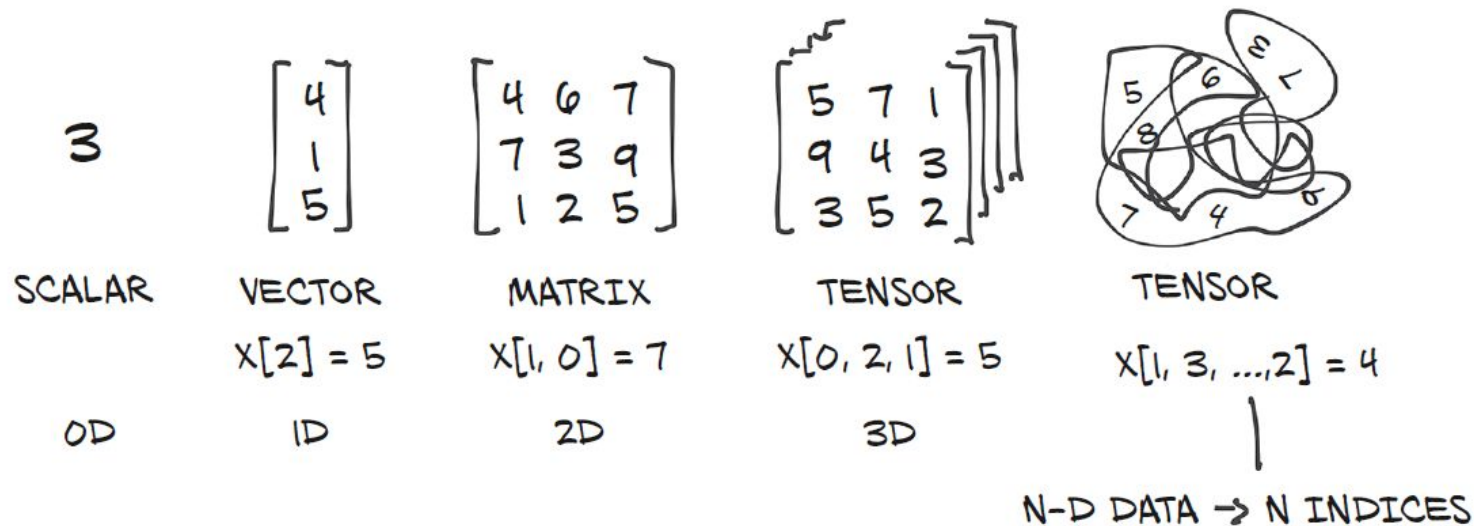


Figure 3.2 Tensors are the building blocks for representing data in PyTorch.

Tensor/tablica vs. lista

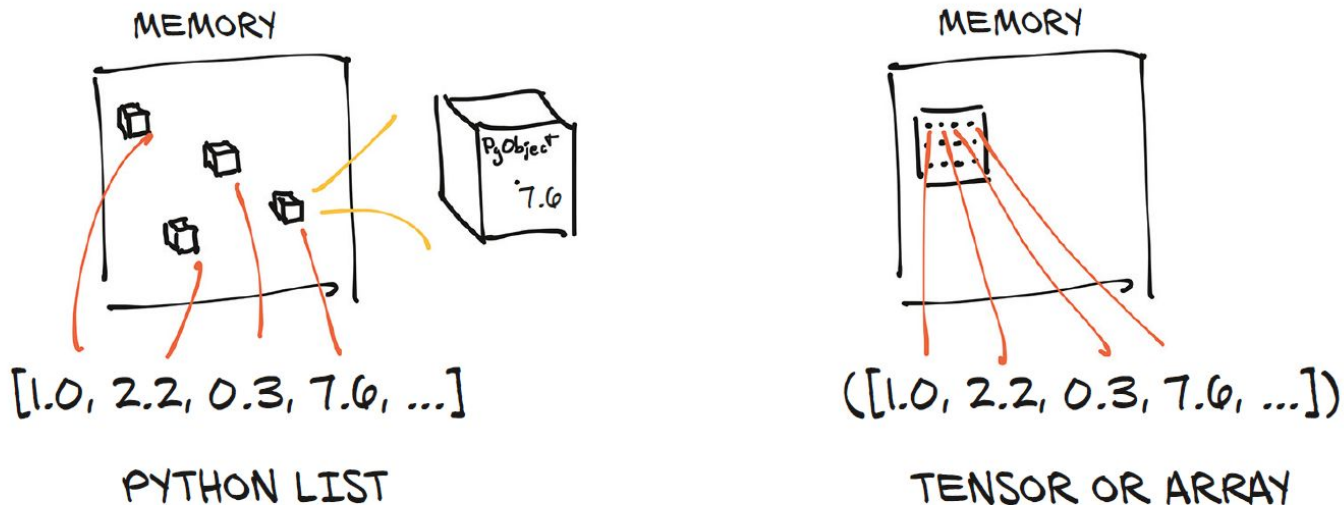


Figure 3.3 Python object (boxed) numeric values versus tensor (unboxed array) numeric values

Arrays

Numpy

mutable

CPU

no automatic
differentiation

numerical

Tensors

Tensorflow

immutable

CPU/GPU/TPU

automatic
differentiation

numerical/strings

Tensory - “widoki” na pamięć

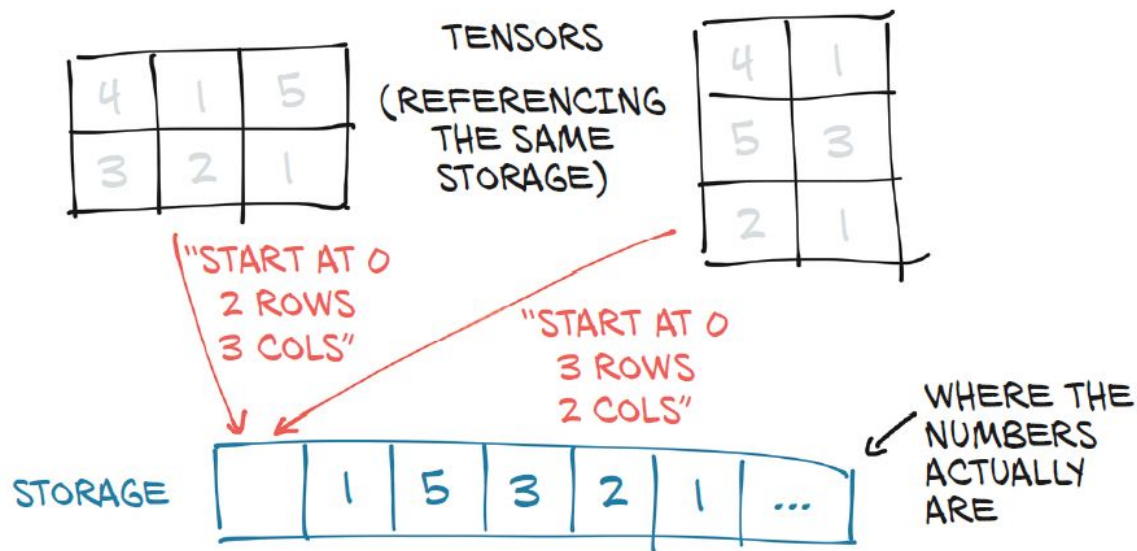


Figure 3.4 Tensors are views of a Storage instance.

Polecane materiały - PyTorch

<https://python.plainenglish.io/numpy-arrays-vs-tensorflow-tensors-95a9c39e1c17>

PyTorch tutorials:

<https://pytorch.org/tutorials/index.html>

Deep Learning with PyTorch

<https://pytorch.org/assets/deep-learning/Deep-Learning-with-PyTorch.pdf>

<https://github.com/borninfreedom/DeepLearning/blob/master/Books/Deep-Learning-with-PyTorch.pdf>

- dla chętnych jako rozszerzenie dzisiejszych zajęć polecam przeczytać rozdział pt.
"It starts with a tensor"
- **większość zajęć będzie opartych na rozdziałach z tej książki**