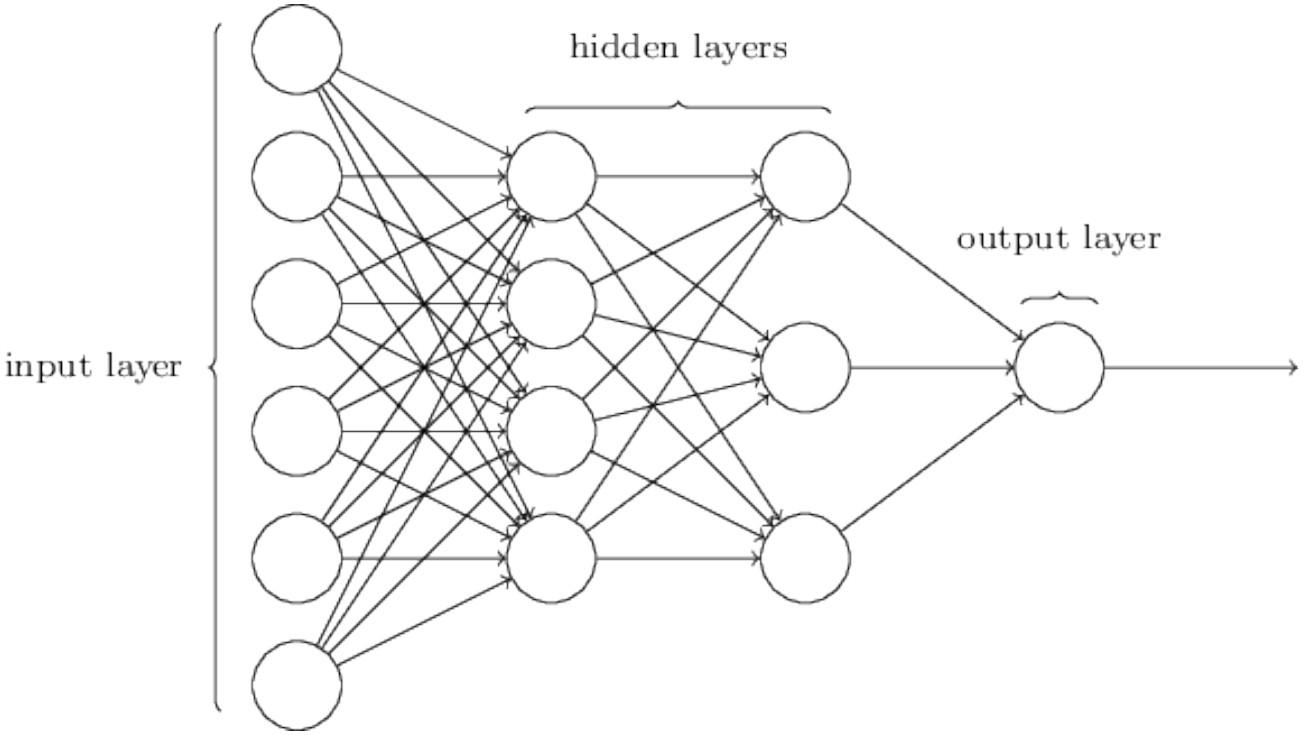
PyTorch Framework

Stanislav Polyakov
SINP MSU

PyTorch (pytorch.org) is an open source machine learning library for Python, based on Torch. It was initially released in October 2016.
PyTorch implements machine learning tools (including convolutions) and supports GPU computations.

Neural networks



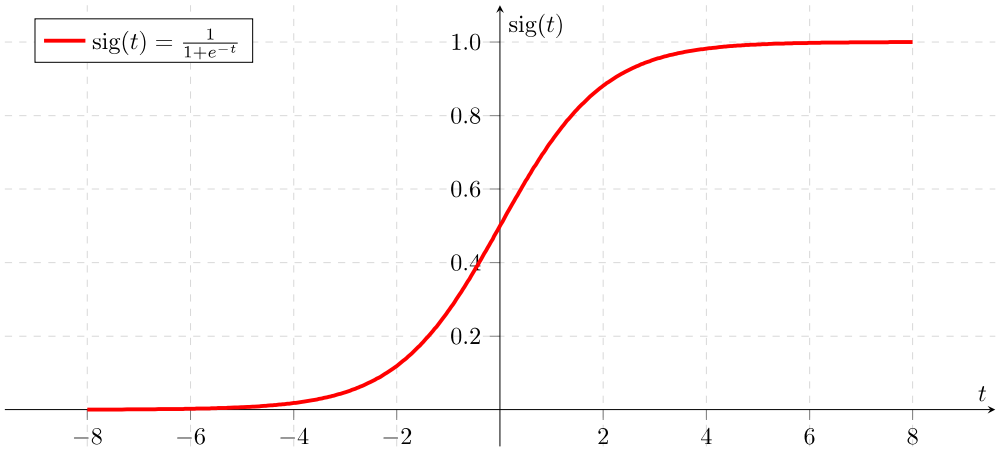
Artificial neural network is a mathematical object represented by nodes, arranged in layers. An output of a node is a result of an activation function applied to a linear combination of input "signals", with weights associated with each input.

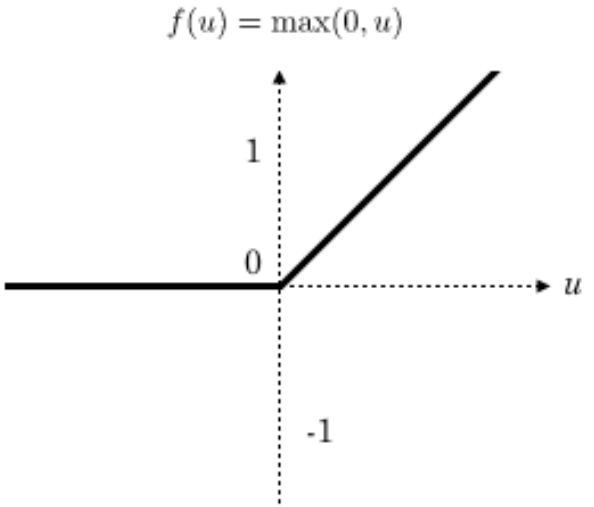
Activation functions

Activation functions are used to introduce nonlinearity.

The choice of a particular activation function may depend on a problem. Typically the differentiable and/or easy to compute functions are used.

Pictured are sigmoid function and rectifying linear unit (ReLU).



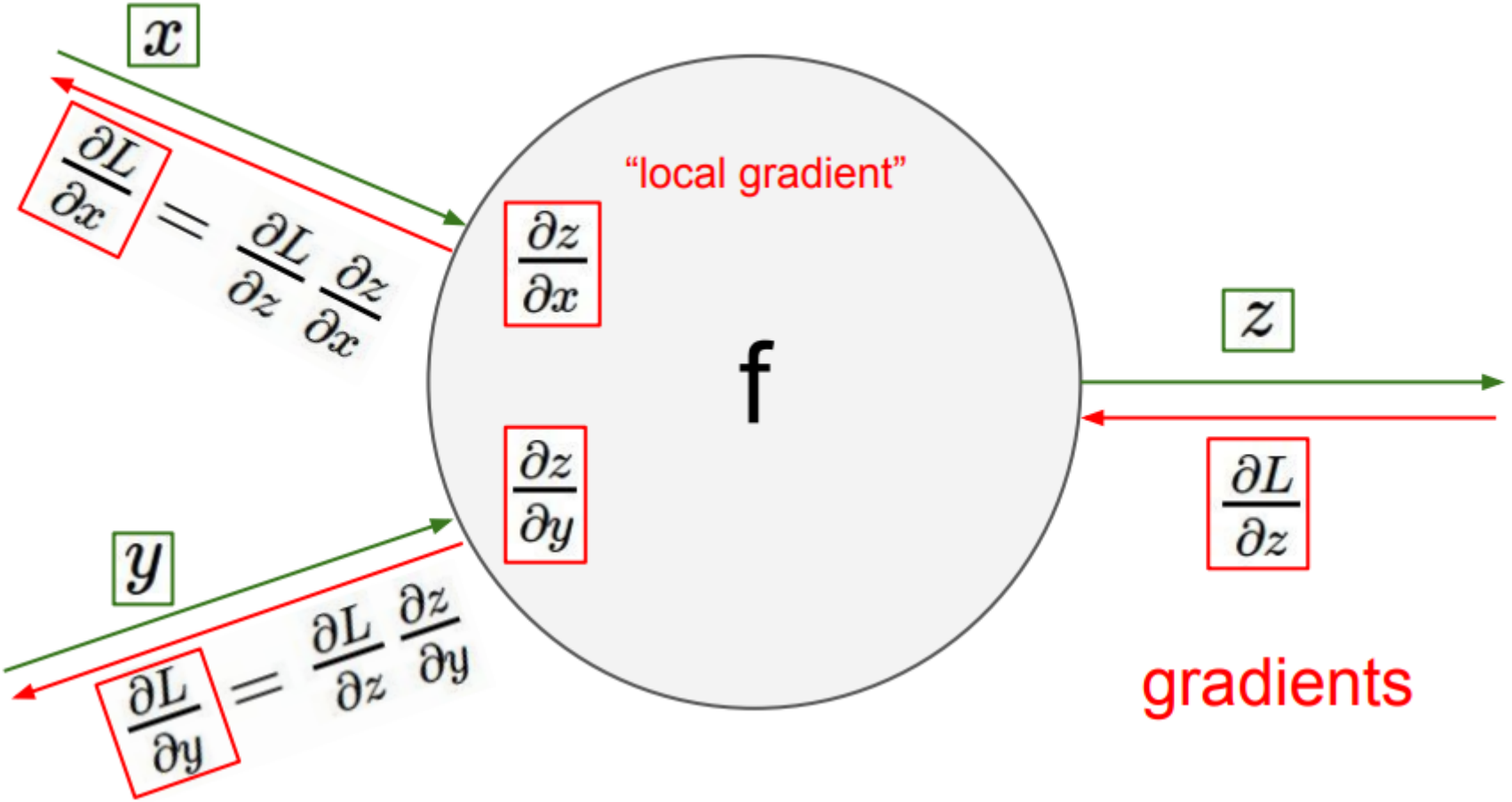


Training Neural Networks

Training a neural network means modifying its weights so that its output gets closer to the desired result. To do that, we take a set of training examples (**x**,**y**) where **x** is the input vector and **y** is the result we want. We use some error or loss function L to measure the distance between the network output N(**x**) and the desired output **y**.

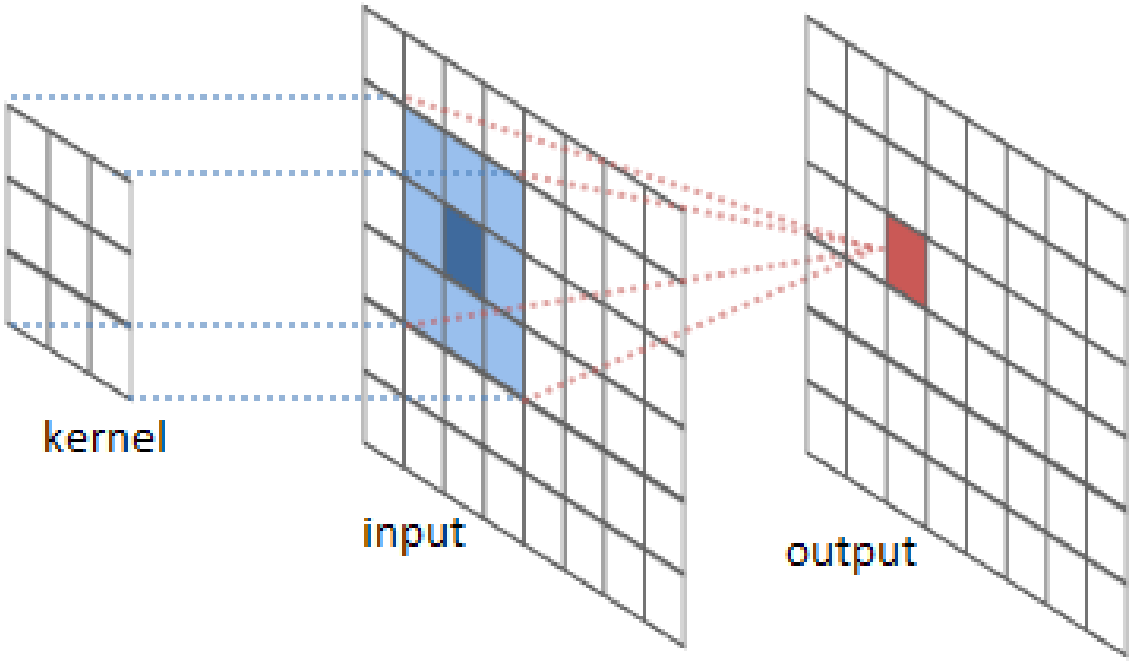
Backpropagation

In a process called backpropagation of errors we find the partial derivatives of L(N(**x**),**y**) with respect to the network's weights and use numerical methods like gradient descent to minimize loss.



Convolutional Networks

Convolutional neural networks have convolutional layers that apply the same operation, or "filter", to the subsets of input ("perceptive fields"), e.g. 3*3 or 5*5 areas.

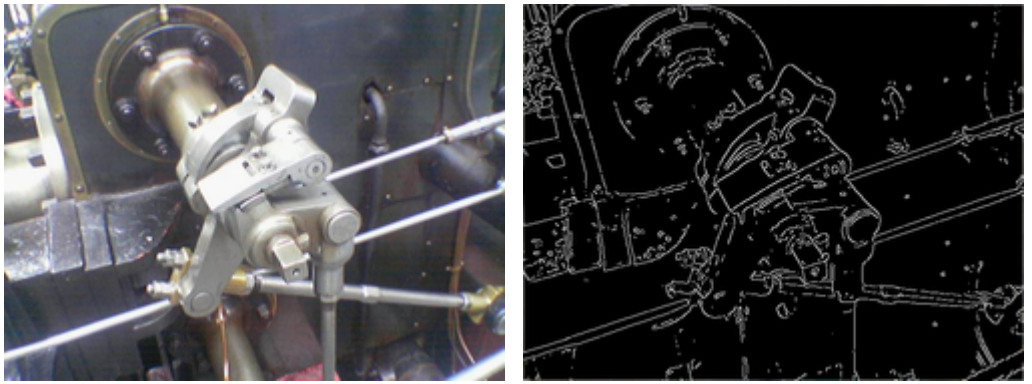


Convolutional Networks

A convolutional layer typically applies multiple filters with learnable weights to the same area and passes their outputs to the next layers.

Convolution helps to solve problems such as image recognition.

(pictured is a result of applying a filter called edge detector)



PyTorch as a Machine Learning Library

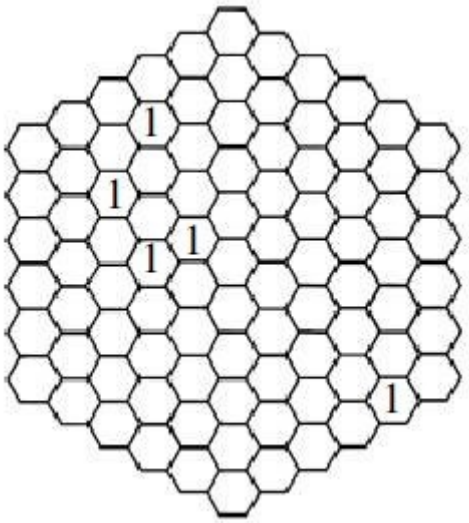
PyTorch has a lot of machine learning and in particular neural network tools implemented. These include: construction elements for neural networks (layers of different types such as fully connected layers and convolutional layers, pooling tools, activation functions etc.) and loss functions. The library supports automatic computation of partial derivatives.

According to the reviews, PyTorch has good performance, is a convenient tool (particularly for those familiar with Python), but may sometimes lack in documentation because it is in early beta.

Pilot Problem

An array of detectors is arranged in hexagonal grid. Each detector only has 2 states (not activated and activated). From 0 to 2 ellipse-shaped areas are chosen randomly. The detector inside an ellipse area has 30% chance to be activated. If the ellipses have common area, the respective detectors have 51% chance to be activated by either of the ellipses. Low-level (1%) random noise is also added.

The problem is to find the number of ellipses, given the states of the detectors.



Hexagonality

Machine learning tools are not designed to handle hexagonal grid. This poses a problem that can be addressed by three approaches:

- Distort the initial hexagonal grid into a rectangular one (shifting every other row by 1/2);
- Overlay the grid with a square grid of virtual detectors and redestribute the data from the actual detectors between them;
- Use or create tools that properly handle the hexagonal grid.

Pilot results

For the pilot problem, I mostly used the distortion approach:

a simple non-convolutional network with 1 hidden layer – 66% correct answers;

a two-layer non-convolutional network – 70% correct answers;

a convolutional network – 83% correct answers.

Overlay grid approach had similar results:

convolutional network – 82%

"Hexagonal approach":

HexagDLy package was used

(computations were several times slower)

convolutional network – 73% correct answers.

Physical problem

560 detectors arranged in hexagonal grid

data for two types of events is generated by the Monte-Carlo algorithm

the task is to recognize the type of the event

first convolutional network – 67% (with bigger training set)