Neural networks and back-propagation explained in a simple way

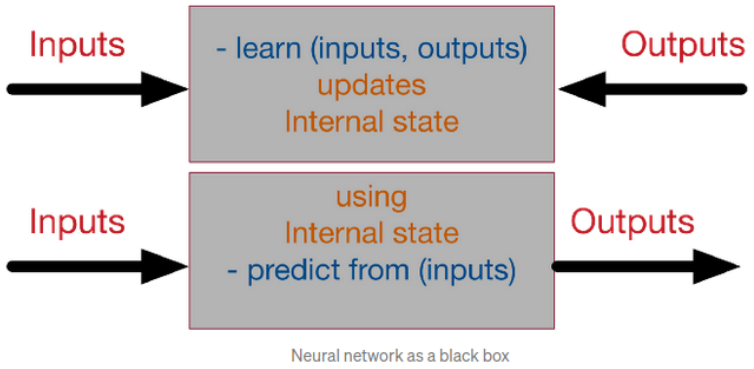
[1]: https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

The learning process:

In → Inputs and desired outputs

Aim → Updates its internal state accordingly. → The calculated output is as close as possible to the desired output.

- The output is is generated according to a training experience. → Machine Learning is called Model Fitting.



Neural network as a black box

Steps:

1/ Model Initialization: Initial Hypothesis: a random initialization.
Next, through the learning process, the models will converge to the ideal solution.
(Example: In linear layers, the model is initialized as follows: y=W.x, where W is the weight of the network and be randomly initialized).

2/ Forward propagate: Check the performance of the model at random.
Input → Neural Network → Output.

3/ Loss Function:
Def: an error metric, which is an indicator on the precision of a NN to reach the desired output from an actual output (also called the error cost function).
Loss = *Absolute value of (desired — actual)*
Loss function = **sum of squares** of the absolute errors

4/ Differentiation
Problematic:
optimization technique to minimize the total loss function, including: genetic algorithms, **greedy search,** a **simple brute-force search**.

	Definition

Brute-Force Search/ Exhaustive Search/ Generate and Test	Enumerate all possible candidates for the solution and check whether each candidate satisfies the problem's statement. [2]
Greedy Search	Problem-solving of making the locally optimal choice at each stage. [3]

[2]: https://en.wikipedia.org/wiki/Brute-force_search
[3]: https://en.wikipedia.org/wiki/Greedy_algorithm

→ brute force: Waste of computational resources and is only used when the model has very few parameters and accuracy is not needed. → Not used in image processing.

Solution: Differentiation Technique (Derivative of the loss function):
* We vary the internal NN weight with a certain small value **δW** → *** how will the total error vary?**
→ **W+ δW** or **W - δW** depending on the derivative of the loss function (<0 or >0).

5/ Back Propagation
Hidden layers between the input and outputs for more variations in the functionality of the neural network.
*Forward-propagate (directly applying the function)
* back-propagate (calculating the derivative of the function)
In back-propagation, we sum the deltas coming from all the target layers.

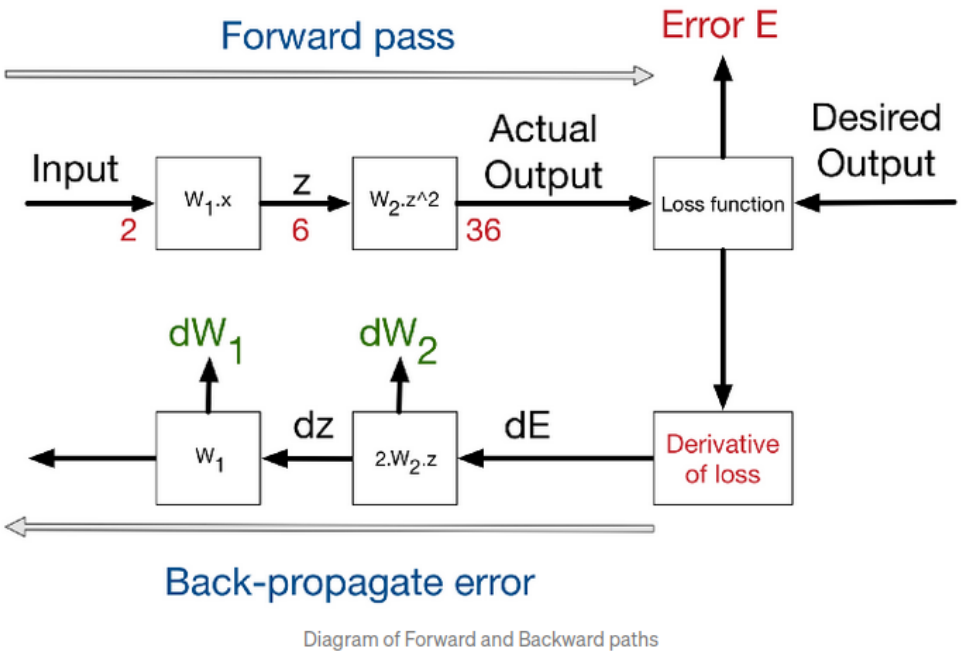


Diagram of Forward and Backward paths

Input → Forward calls → Loss function → derivative→back-propagation of errors.

6/ Weight Update
The delta rule:
New weight = old weight — Derivative * learning rate ; with the learning rate having very small.

*Best Practice:

Update the weights every **Batch Size=N** of training → mini-batch gradient descend.

N=1 → Full online-learning or stochastic gradient descent.

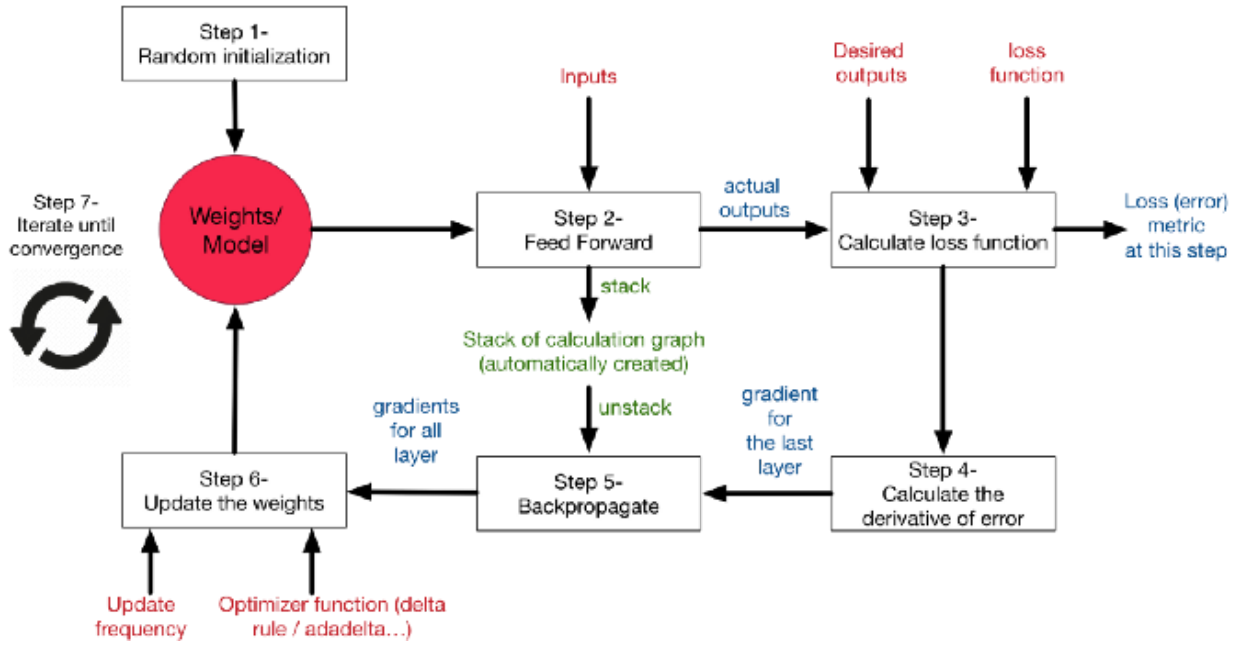
7/ Iterate until Convergence

High learning rate means faster learning, also higher chance of instability.
It depends on the random initialization of the network.
It depends on the quality of the training set.

→ What we need to learn:

The execution of NN has 4 major steps: [4]

1. Forward step (where we go from inputs to outputs)
2. Loss function (where we compare the calculated outputs with real outputs)
3. Backward step (where we calculate the first delta at the loss function and then back-propagate)
4. Optimization step (where we update the internal weights with deltas and learning rate)



[4]
https://medium.com/datathings/a-neural-network-fully-coded-in-numpy-and-tensorflow-cc275c2b14dd

Writing a simple neural network from scratch in C, without the help of any vector or matrix libraries:
powerful math libraries → numpy.