



# A Primer on Neural Networks

Andrea Esuli

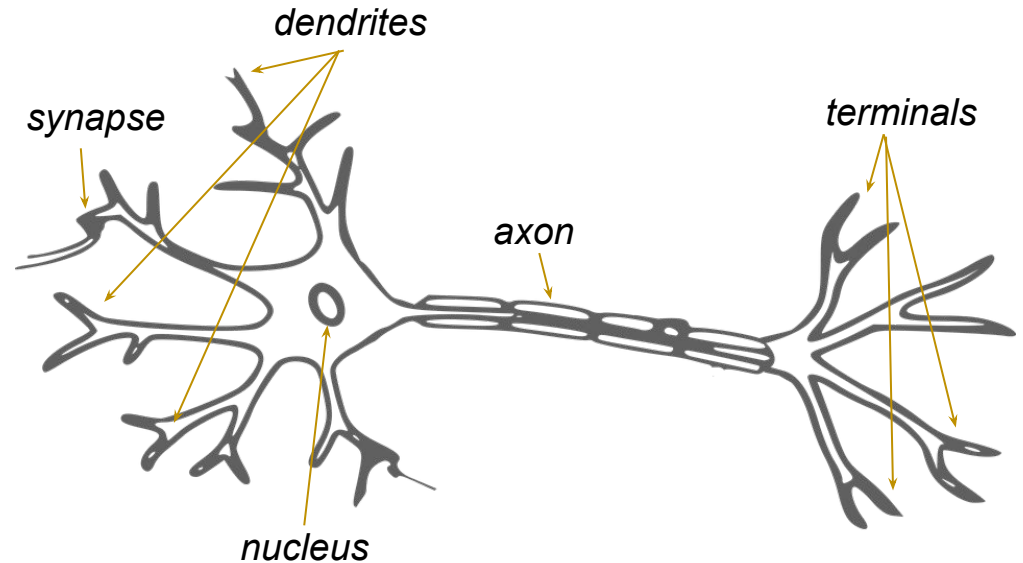


# The origins

Neural networks (NN) are called "Neural" because their first formulation was inspired to the biological structure of the brain of animals.

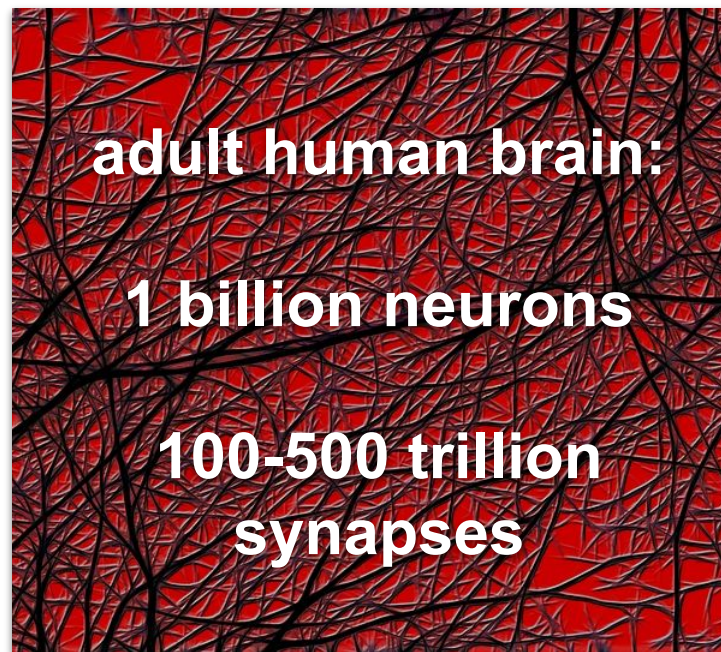
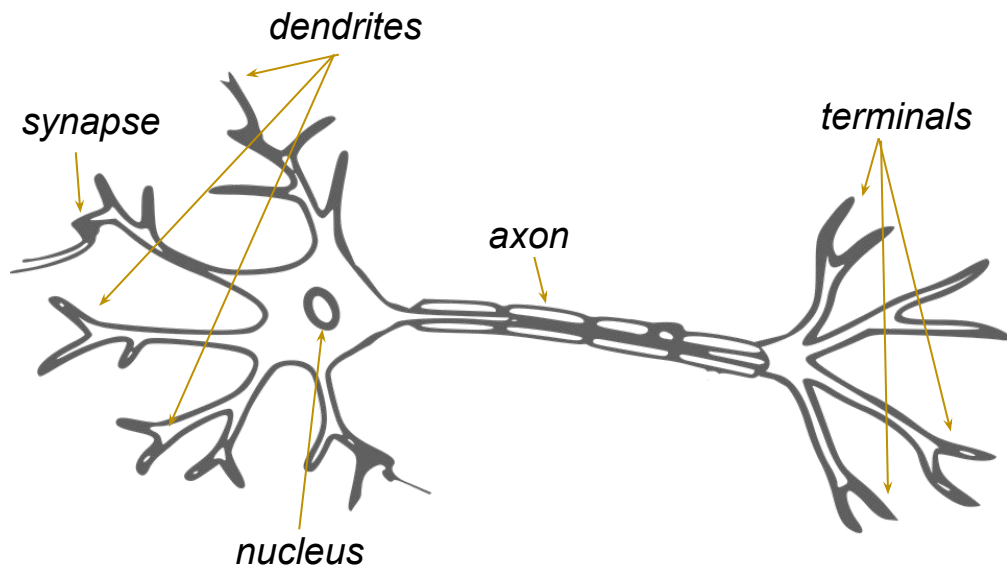
*Dendrites* collect *signals* from *terminal* of other neurons that are connected by *synapses*.

Depending on the collected signals a neuron can itself send a signal to other neurons through its *axon*.



# The origins

Neural network (NN) are called "Neural" because their first formulation was inspired to the biological structure of the brain of animals.



# The origins

An artificial neuron:

- takes in input a vector of values
- combines them **linearly** with a **weighted sum (pre-activation)**
- fires a signal using a **non-linear activation function  $f$**

$$o = f(\sum x_i w_i)$$

The **parameters of the model** (that are fit at learning time) are thus the vector of  $|i|$  weights (one for each of the  $|i|$  input values).

The activation function is a **non-linear transformation**.

# Neural Network

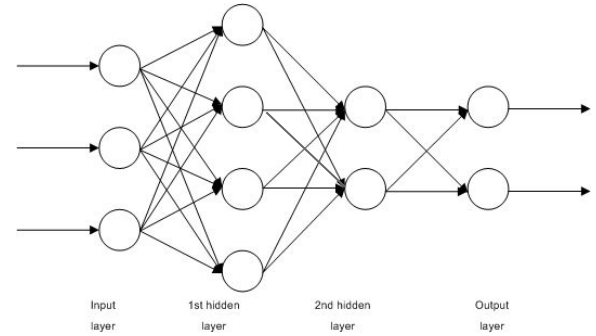
A **layer**\* in a network is a set  $|j|$  of neurons, i.e., a  $|i| \cdot |j|$  **matrix of weights (and biases)**.

It produces as output a vector of length  $|j|$ .

The output of a layer is passed to its activation function and the result become the input of the next layer (or the output if the layer is the last one).

***Without the nonlinearity introduced by the activation function, the network would collapse into a simple linear transformation.***

\*This is a *dense layer*, also called *fully-connected* layer. Other type of layers exist as we will see later.



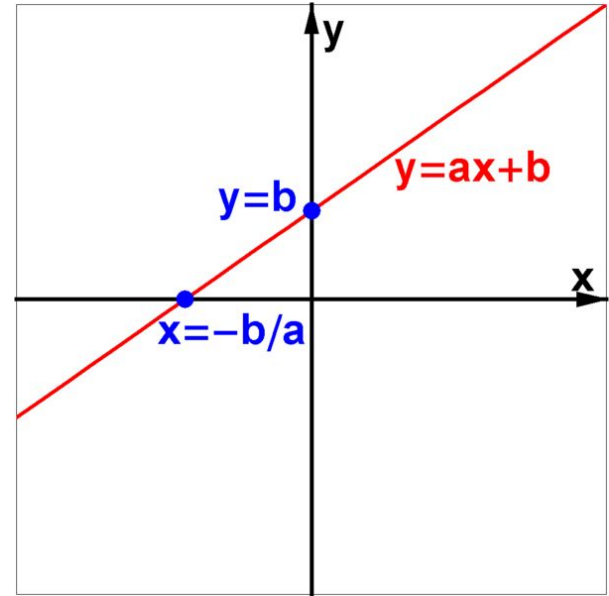
# Bias

A neuron usually have a **bias** value, i.e., a constant value that is added to pre-activation.


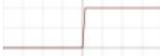


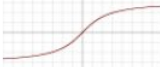
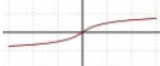
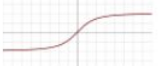



Bias is not constant with respect to the learning process, it is a **parameter** that is fitted exactly like all the others.

Bias can be seen as the weight of an additional input that is constantly one.

Bias enables to **offset** the (linear) pre-activation value with respect to an all-zero input, similarly to the intercept value of the line equation.



# Activation functions

Name	Plot	Equation	Derivative (with respect to $x$ )	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$
Softsign		$f(x) = \frac{x}{1 +  x }$	$f'(x) = \frac{1}{(1 +  x )^2}$	$(-1, 1)$
Inverse square root unit (ISRU)		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left( \frac{1}{\sqrt{1 + \alpha x^2}} \right)^3$	$(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}})$
Rectified linear unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$
Parameteric rectified linear unit (PReLU)		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Exponential linear unit (ELU)		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$

# Forward propagation

The **forward propagation** (forward pass) is the process that transforms some input data elaborating it through the levels of a neural network until some output is produced.

Forward pass on a neural network with a two-neurons hidden layer with relu activation and a single-neuron output layer with sigmoid activation.



# Training a Neural Network

Weights in the matrix are initialized randomly (or with pre-trained values).  
Weights must differ to break symmetry.

Training data is passed into the network (*forward pass*).

The output of the network is compared with the expected output (true label in training data), computing the **error** the network made with respect to a **loss function**.

Correction is made by means of backpropagation and gradient descent, changing each weight so as to reduce the error.

# Gradient descent

The idea of gradient descent applied to NNs is pretty simple:

- The derivative  $df(x)/dx$  of a function  $f(x)$  indicates its slope.
  - $df(x)/dx > 0$  means that a **local** increase/decrease of  $x$  increases/decreases  $f(x)$
  - $df(x)/dx < 0$  means that a **local** increase/decrease of  $x$  decreases/increases  $f(x)$
- If we take as  $f(x)$  the **error function** (loss) of the network and as  $x$  a **weight of the network**, by computing the derivative we can determine how to **change the weight so as to decrease the error**.

For this to work it is necessary that all involved computations are **differentiable**.

# Backpropagation

Computing the derivative of all the parameters of a deep network can be made efficient by

- exploiting the [chain rule](#) of derivatives and
- backpropagating gradients, i.e., starting the computation of gradients close to the error function and reusing such computations for gradients of elements that are more distant (thus navigating the network backward).

[Step by step training of a network with backpropagation.](#)

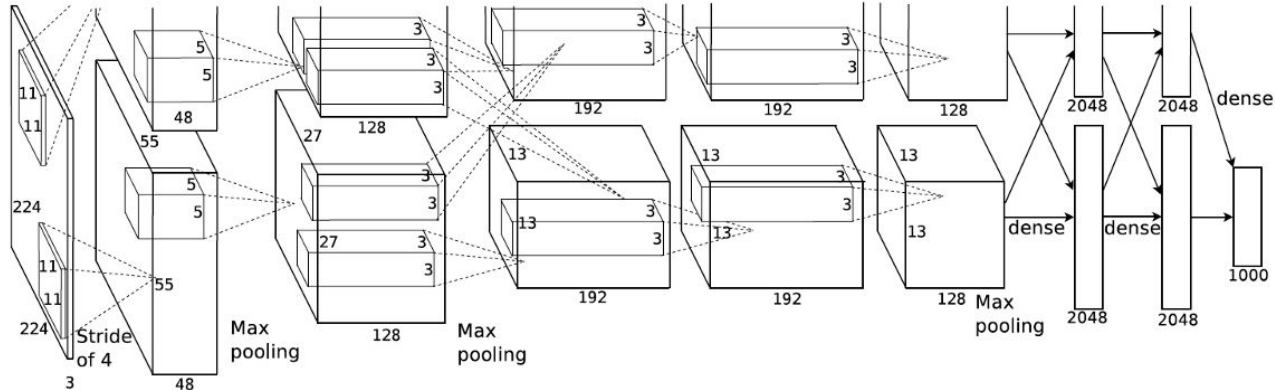
[Visual demo](#)

# Deep Learning

In 2012 [AlexNet](#) won the ImageNet image classification competition by a large margin, by using NN.

Their network was a very large (and deep, for the standard of the time) one.

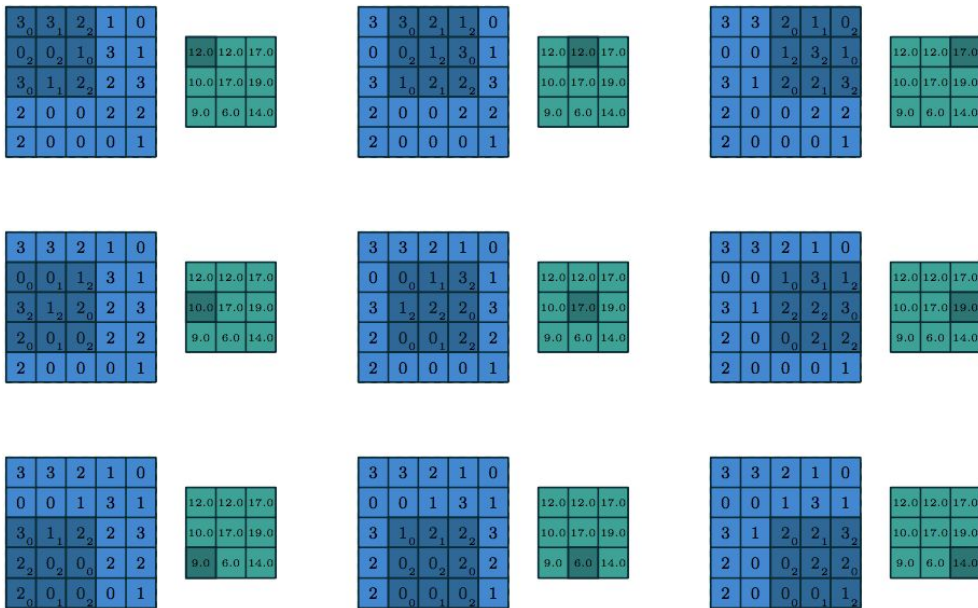
It revived NNs exploiting to two new factors: **Big data** and **GPU**.



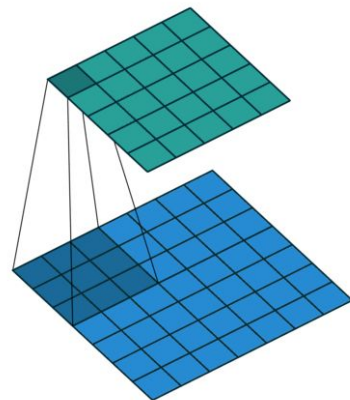
# Convolutional Layer

A convolutional layer in a NN is composed by a set of **filters**.

- A filter usually has a many dimensions as the data type it is applied to.
  - Images use 2D filters, text 1D.
- A filter combines a "local" selection of input values into an output value.
- All filters are "swept" across all input.



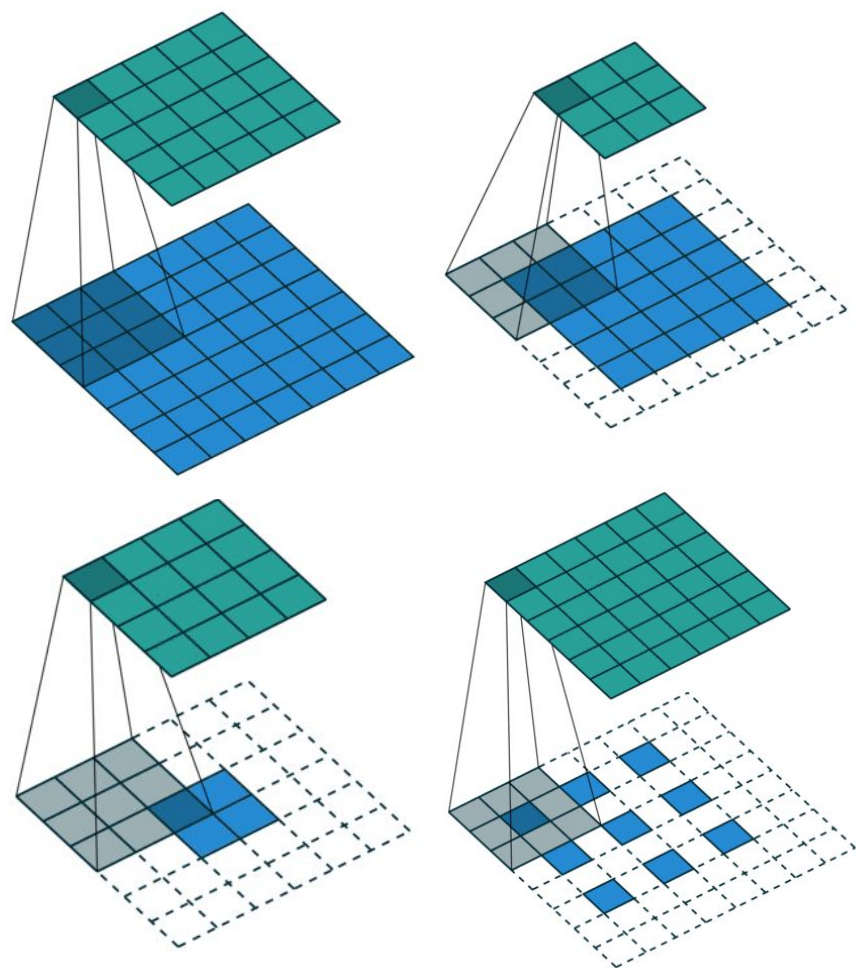
0	1	2
2	2	0
0	1	2



[Images from "A guide to convolution arithmetic for deep learning" Vincent Dumoulin, Francesco Visin](#)

# Convolutional Layer

- Filters have additional parameters that define their behavior at the start/end of documents (padding), the size of the sweep step (stride), the eventual presence of holes in the filter window (dilation).
- During training each filter specializes into recognizing some kind of relevant combination of features.
- CNNs work well on stationary feats, i.e., those independent from position.

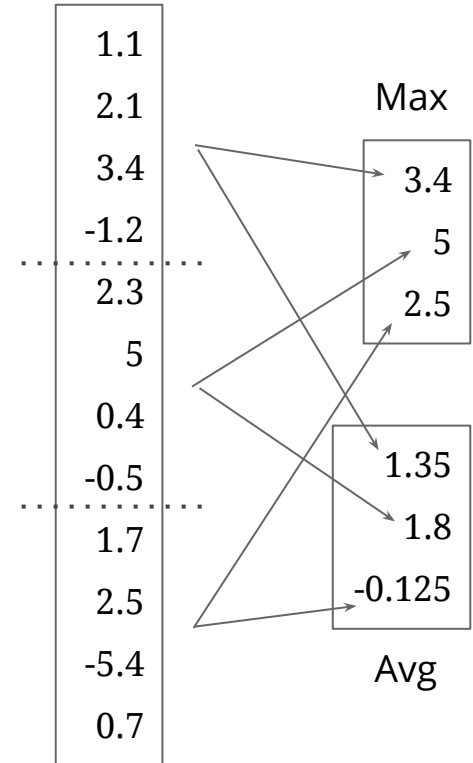


[Images from "A guide to convolution arithmetic for deep learning" Vincent Dumoulin, Francesco Visin](#)

# Pooling

A **pooling** layer aggregates (max, average) output of groups of units into a single value for the next layer.

- It reduces the number of parameters of the model (downsampling)
- It contrasts overfitting.
- It add robustness to local variations (translation)
- It is used to convert **variable length inputs**, e.g., from **CNNs**, to the have the **fixed length**, thus enabling connecting segments of networks that produce output of variable size to layers with fixed-size input.



# Dropout

A **dropout** layer randomly hides output of units from a layer to the next.

- It is a regularization technique that contrasts **overfitting** (i.e., being too accurate on training data and not learning to generalize).
- It can also help breaking cases of symmetry in the network.

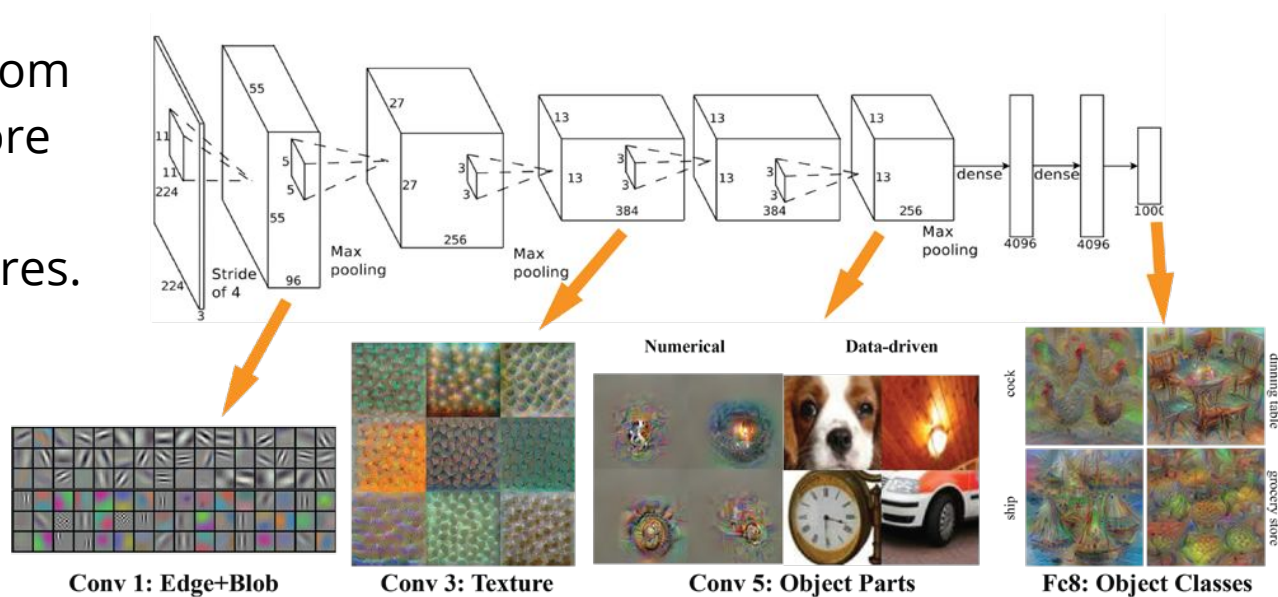




# Convolutional Neural Network

CNNs have been successfully applied on images.

- First level of a stack of CNNs capture local pixel features (angles, lines)
- Successive layers combine features from lower levels into more complex, less local, more abstract features.



[\[image source\]](#)

# Recurrent Neural Networks

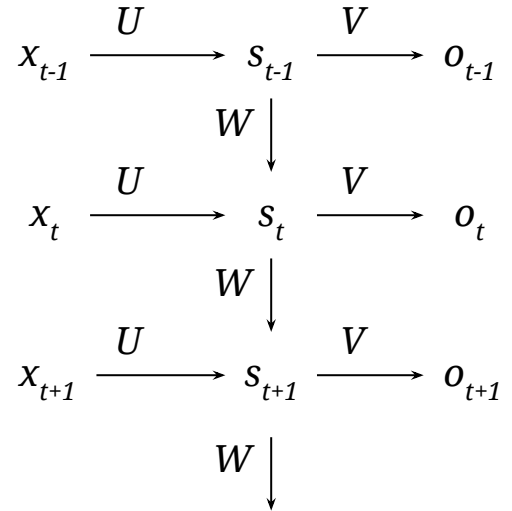
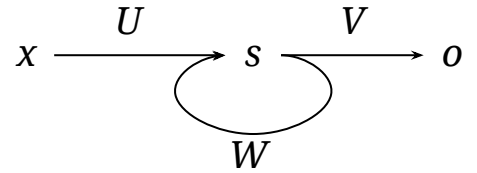
A Recurrent Neural Network (RNN) is a neural network in which connections between units form a directed cycle.

Cycles allow the network to have a memory of previous inputs, combining it with current input.

RNNs are fit to process sequences, such as text.

Text can be seen as a sequence of values at many different levels: characters, words, phrases...

[Suggested read](#)

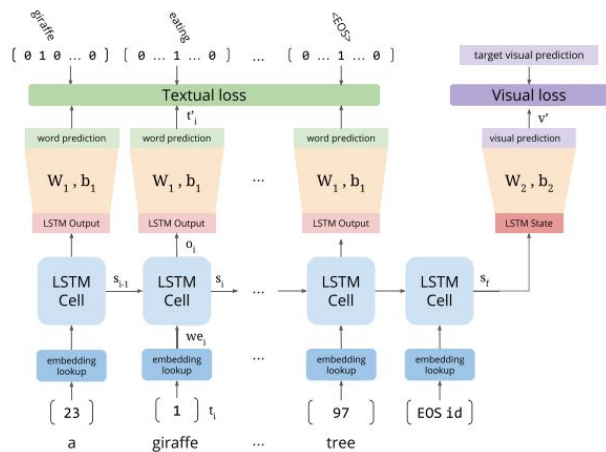
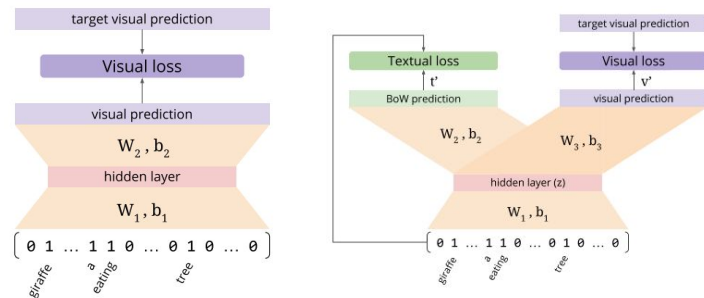


# From feature engineering to network engineering

NN-based learning frees the knowledge engineer from the burden of feature engineering.

The layers in the NN can implicitly learn abstract, high-level, semantic representation from the raw input data.

But... the network has to be properly designed in order to be able to perform the assigned task, and the possible NN configurations for a task are endless.



# Software 2.0

Andrej Karpathy, Director of AI at Tesla.

*"The "classical stack" of Software 1.0 is what we're all familiar with...  
...It consists of explicit instructions to the computer written by  
a programmer...."*

*In contrast, Software 2.0 **is written in neural network weights.**"*

*"In the case of neural networks, we restrict the **search** to a continuous subset **of the program space** where the search process can be made (somewhat surprisingly) efficient with **backpropagation** and **stochastic gradient descent.**"*



# Software 2.0

Andrej Karpathy, Director of AI at Tesla.

*"**Software 2.0 is not going to replace 1.0** (indeed, a large amount of 1.0 **infrastructure** is needed... ..), but it is going to take over increasingly large portions of what Software 1.0 is responsible for.."*



*"Visual Recognition... Speech Recognition... Machine Translation... Games... Robotics... **Databases**..."*

# Deep Learning est mort...

Yann LeCun, *Director of Facebook AI Research*

*"...the important point is that people are now building **a new kind of software** by assembling networks of parameterized functional blocks and by training them from examples using some form of **gradient-based optimization**."*



# ...Vive Differentiable Programming

Yann LeCun, Director of Facebook AI Research

*"An increasingly large number of people are **defining the networks procedurally** in a data-dependent way (with loops and conditionals), allowing them to **change dynamically** as a function of the input data fed to them. It's really **very much like a regular program**, except it's **parameterized, automatically differentiated, and trainable/optimizable...**"*

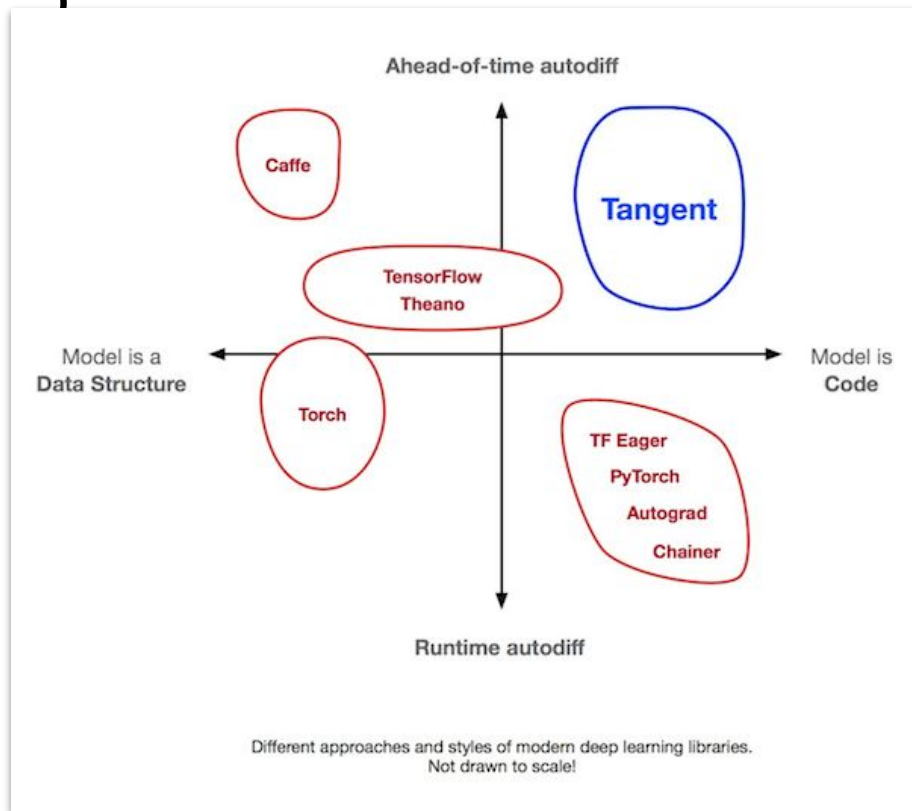


# Differentiable Programming

The backpropagation-based model of fitting differentiable functions is moving beyond NN.

Any code that is differentiable can be seen as a model that can be fitted by back propagation.

Google Tangent: source-to-source debuggable derivatives in pure python.





# Differentiable Programming

The backpropagation-based model of fitting differentiable functions is moving beyond NN.

Any code that is differentiable can be seen as a model that can be fitted by back propagation.

Google Tangent: source-to-source debuggable derivatives in pure python.

## Source Code Transformation with Tangent: Conditionals

```
def f(x):  
    if x > 0:  
        a = x ** 2.0  
    else:  
        a = -x  
    return a
```