Software development

Section 3: Neural Networks

Ioannis Emiris and Emmanouil Christoforou

Dept Informatics & Telecoms, NKUA

December 15, 2019

Introduction

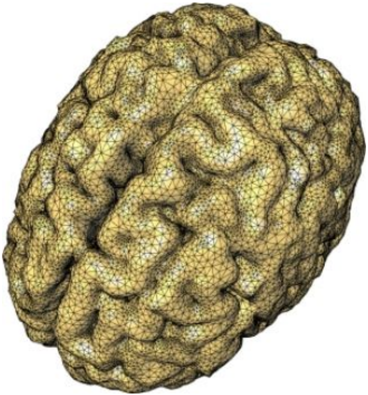
Machine Learning

Machine learning (ML) uses statistical techniques to give computer systems the ability to ``learn'' (i.e. progressively improve performance on a specific task) from data, without being explicitly programmed.

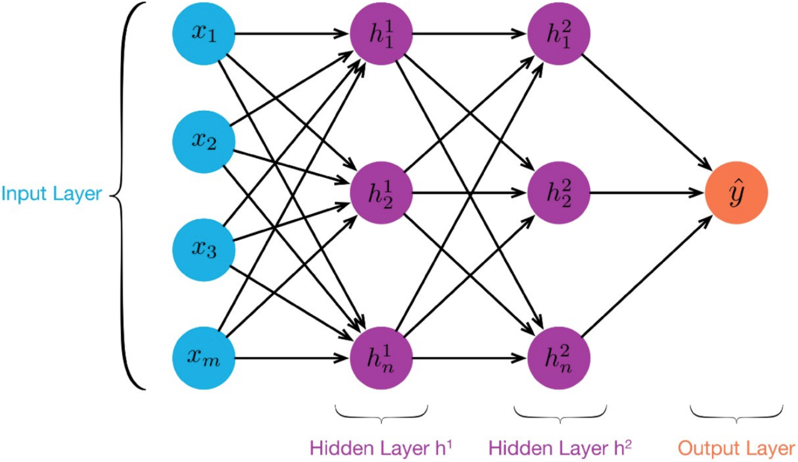
Neural Networks

Neural Networks (and Deep Learning) is a class of ML algorithms that use a cascade of layers of nonlinear processing units for feature extraction and transformation, mimicking the human neurons. Each successive layer uses the output from the previous layer as input.

Live Neural Network

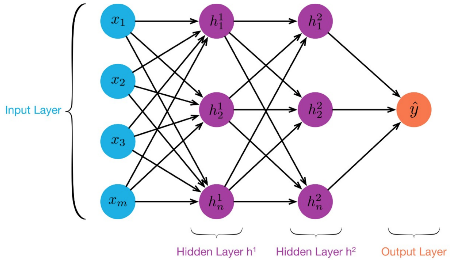


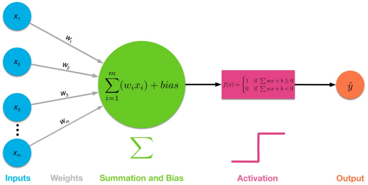
Artificial Neural Network



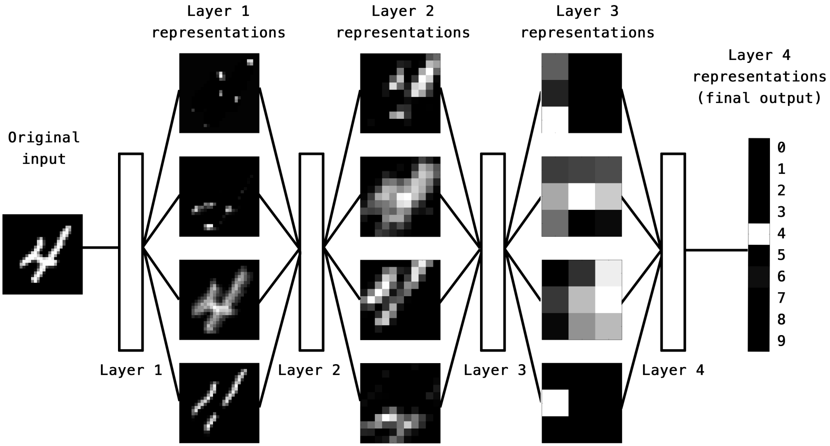
Neural Networks

- NN is a cascade of **hidden layers**
- Each layer is a **data transformation** function
  - weights/parameters determine the transformation
  - transformations are differentiable for improving output





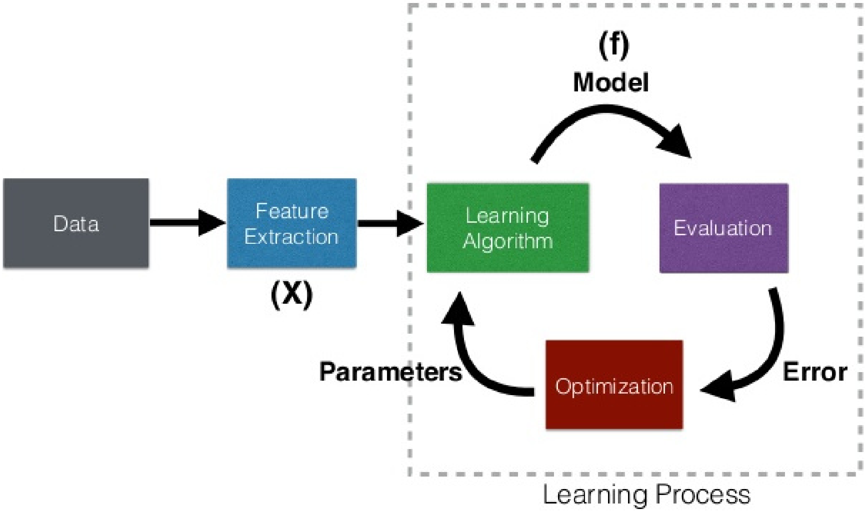
NN Layers of representations



- Input: Raw data
- Data transformed so that **irrelevant** information is **filtered out**

Outline

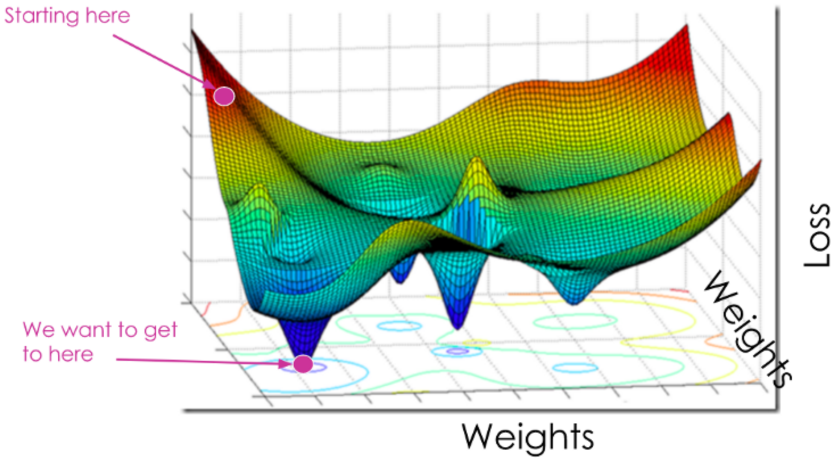
Learning process



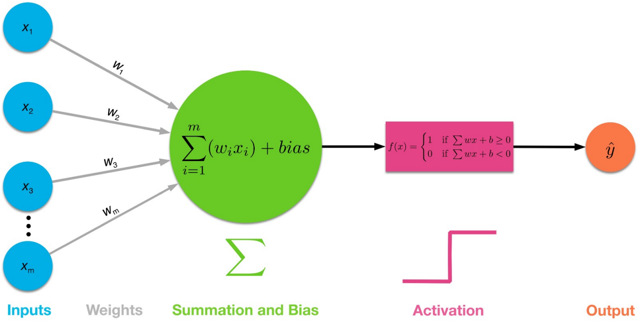
Predicted $f(X)$: Find f s.t. $f(x)$ is close to **True value**

Goal of training

Goal of training $\rightarrow$ find **weights** that minimise loss function

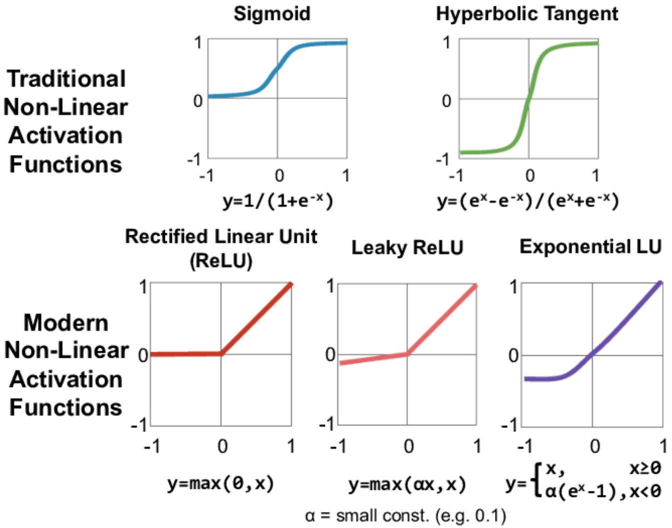


Activation functions



- **Non-linear activation** transformation on (weighted) input
- Generate **non-linear mappings** from inputs to outputs
- Decide whether this neuron is **active** or not

Activation functions (I)



Loss functions based on problem

y = true output label/value, $\widehat{y}$ = predicted class/value,
n = number of training instances, M = #classes.

- Regression

Mean Squared Error (MSE): $$\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

Mean Absolute Error (MAE): $$\frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|$$

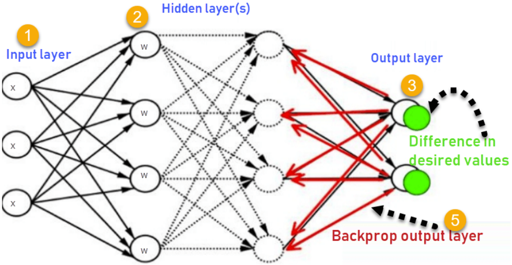
- Classification

binary cross-entropy: $$-\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log \widehat{y_i} + (1 - y_i)\log(1 - \widehat{y_i})\right)$$

(multiclass, one label) cross-entropy: $$-\frac{1}{n}\sum_{i=1}^{n}\sum_{j}^{M} y_{ij} \log \widehat{y_{ij}}$$

Backward propagation

- Forward propagation: compute activations (black arrows)
- Backprop: compute derivatives optimizing parameters/weights (red)
- Gradient descent exploits chain rule $\frac{dL}{dx} = \frac{dL}{dw}\frac{dw}{dx}$
- . . . to get **partial derivative of loss** wrt **weights**
- Learning rate: step to change weights



Overfitting

NN as universal approximators

Universal Approximation Theorem

Let ξ be a non-constant, bounded, monotonically-increasing continuous activation function, $f : [0, 1]^d \to \mathbb{R}$ continuous, and $\epsilon > 0$. Then, $\exists n$, parameters $a, b \in \mathbb{R}^n$, $W \in \mathbb{R}^{n \times d}$ s.t.

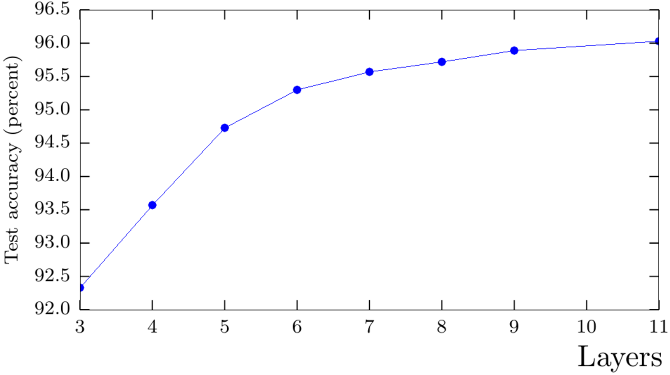
$$\left| \sum_{i=1}^{n} a_i \xi(w_i^T x + b_i) - f(x) \right| < \epsilon.$$

- Any f approximated arbitrarily well by NN with one hidden layer
- How many neurons? How to find the parameters?
- Does it generalize well? Does it overfit?
- Shallow nets work poorly for high-dimensional data like images

Cybenko 1989; Hornik 1991

Better Generalization with Greater Depth

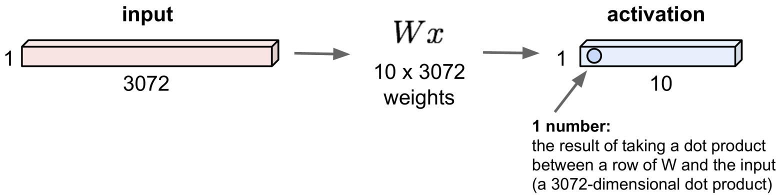
- Empirical results show that **deeper** networks **generalize** better
- Test accuracy consistently increases when increasing depth



Transcribe multidigit numbers from photographs of addresses (Goodfellow'14)

Outline

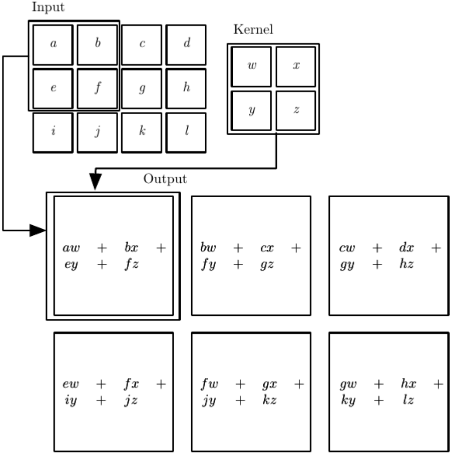
Fully Connected Layer



Matrix multiplication: W is a 10×3072 real matrix.

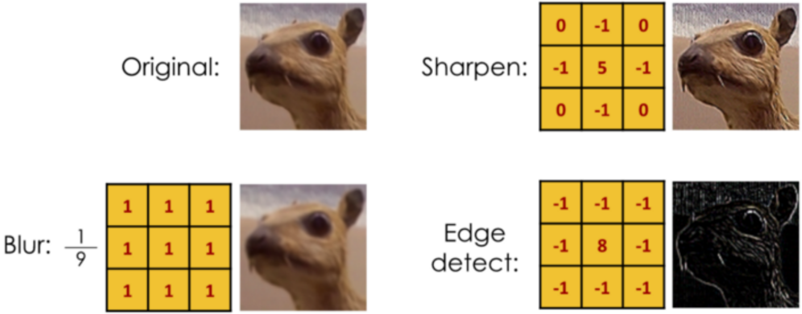
Convolution definition

- Pointwise matrix multiply
- The kernel (convolution mask) is a small matrix (or flipped)



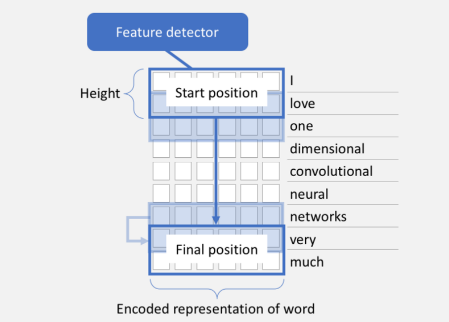
Convolution

Motivation: Sparse / Parameter sharing / Equivariant representations



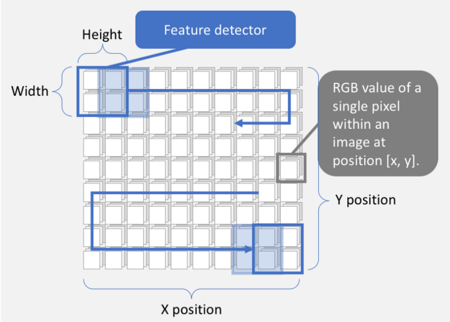
1D vs 2D convolution

1D Convolutional - Example



In this example for natural language processing, a sentence is made up of 9 words. Each word is a vector that represents a word as a low dimensional representation. The feature detector will always cover the whole word. The height determines how many words are considered when training the feature detector. In our example, the height is two. In this example the feature detector will iterate through the data 8 times.

2D Convolutional - Example



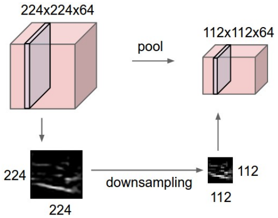
In this example for computer vision, each pixel within the image is represented by its x- and y position as well as three values (RGB). The feature detector has a dimension of 2 x 2 in our example. The feature detector will now slide both horizontally and vertically across the image.

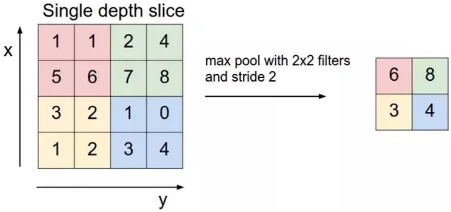
---

https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-t

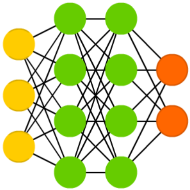
Pooling (Downsampling)

- Makes representations smaller and more manageable
- Helps make representation roughly invariant to small input translations
- **Max**, Avg, L^2 norm, weighted avg distance from the central pixel



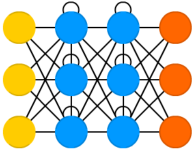


Recurrent Neural Networks (RNN)



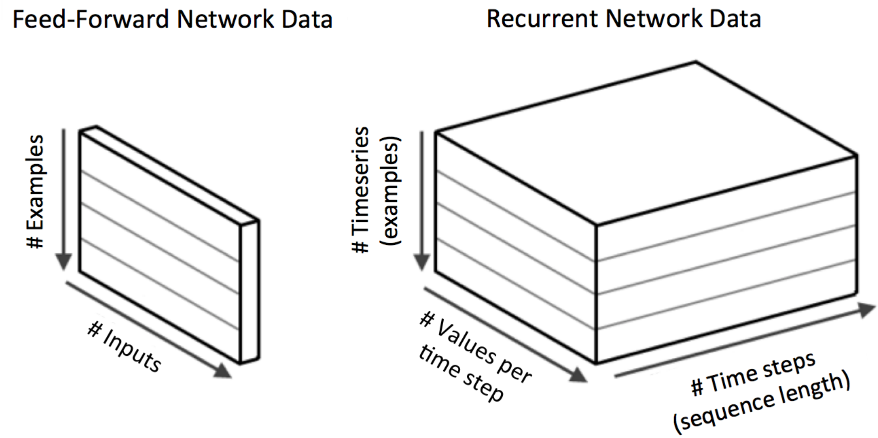
NNs mainly distinguished in:

- Feedforward neural networks (FNN):
  Features processed independently
- Recurrent neural networks (RNN):
  Features processed sequentially e.g. time series

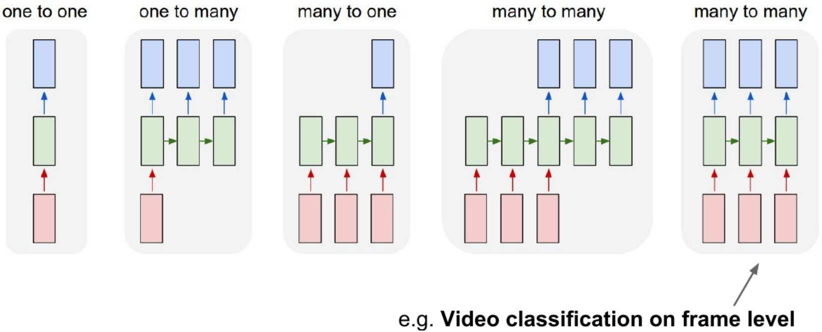


RNN allow **cyclic** connections
$\rightarrow$ fit best to process **sequential** data

RNN input


Feed-Forward Network Data
Examples
Inputs
Recurrent Network Data
Timeseries
(examples)
Values per
time step
Time steps
(sequence length)

RNN: Type of sequences



1-many: image captioning (image $\rightarrow$ word sequence)

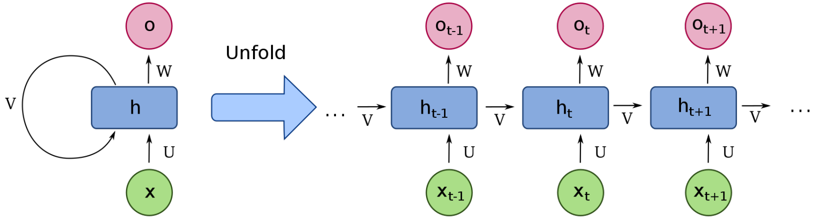
many-1: sentiment classification (word sequence $\rightarrow$ sentiment)

many-many: machine translation (word sequence $\rightarrow$ word seq.)

many-many: video classification on frame level

Recurrent neural networks

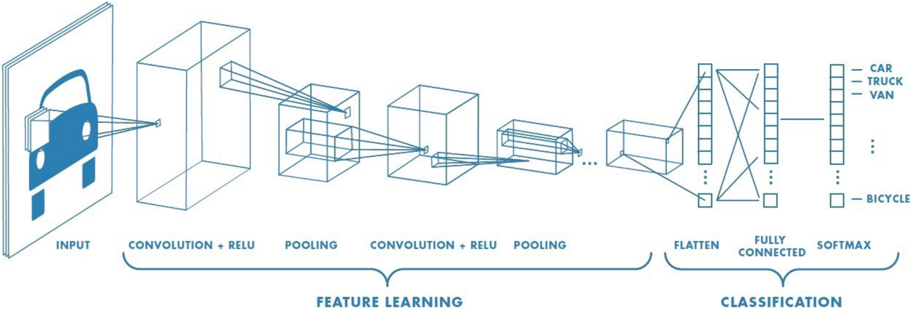
- allow cyclic connections
- . . . essentially offer internal state (memory)
- keep track of arbitrarily long-term dependencies (boon/issue)
- internal state processes input x_t's by applying recurrence formula at every time step t: new state $h(t) \leftarrow f_w(h_{t-1}, x_t)$.



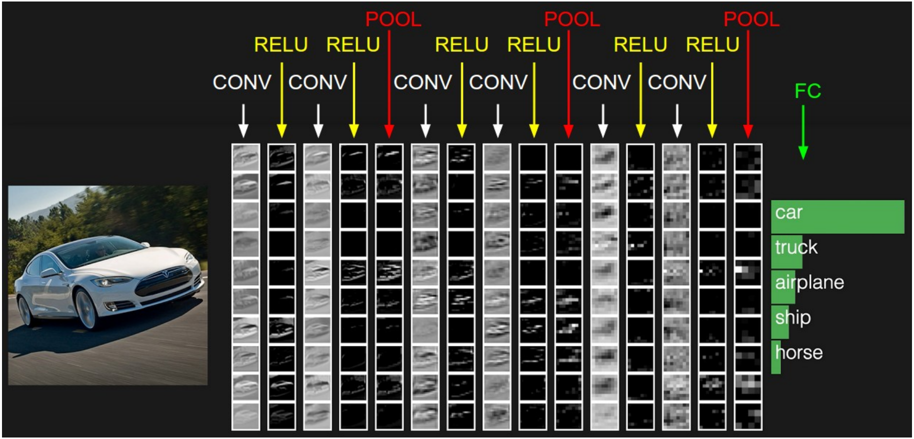
Numerics: Back-propagated gradients may vanish ($\rightarrow 0$) or explode ($\rightarrow \infty$).

Outline

Typical Neural Network



CNN example


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Development frameworks

Caffe (UC Berkeley) → Caffe2 (Facebook)

Torch (NYU / Facebook) → **PyTorch (Facebook)**

Theano (U Montreal) → **TensorFlow (Google)**

PaddlePaddle (Baidu)

Chainer

MXNet (Amazon)
Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

CNTK (Microsoft)

JAX (Google)

And others...

Training may be done on GPU environment Colab (Google).

References

- Deep Learning book (https://www.deeplearningbook.org/) (www.dropbox.com/s/oj3olyzxvnchrqs/SGP%202018.pdf?dl=0)
- Yannis Avrithis, Deep learning for computer vision (6h short course) (http://erga.di.uoa.gr/meetings/Slides_Avrithis.zip)
- J.J. Allaire, Machine Learning with TensorFlow and R (https://beta.rstudioconnect.com/ml-with-tensorflow-and-r/)