# Introduction to Deep Learning

**Copernicus Master on Digital Earth**

Lecture 1: Success, challenges and basic concepts

Prof. Nicolas Courty
ncourty@irisa.fr

# Today

Set the fundamentals of machine learning.

- Why learning?
- Applications and success
- Statistical learning
  - Supervised learning
  - Empirical risk minimization
  - Under-fitting and over-fitting
  - Bias-variance dilemma

# Why learning?

What do you see?

*How do we do that?!*

Sheepdog or mop?

Credits: Karen Zack, 2016.

Chihuahua or muffin?

Credits: Karen Zack. 2016.

The automatic extraction of semantic information from raw signal is at the core of many applications, such as

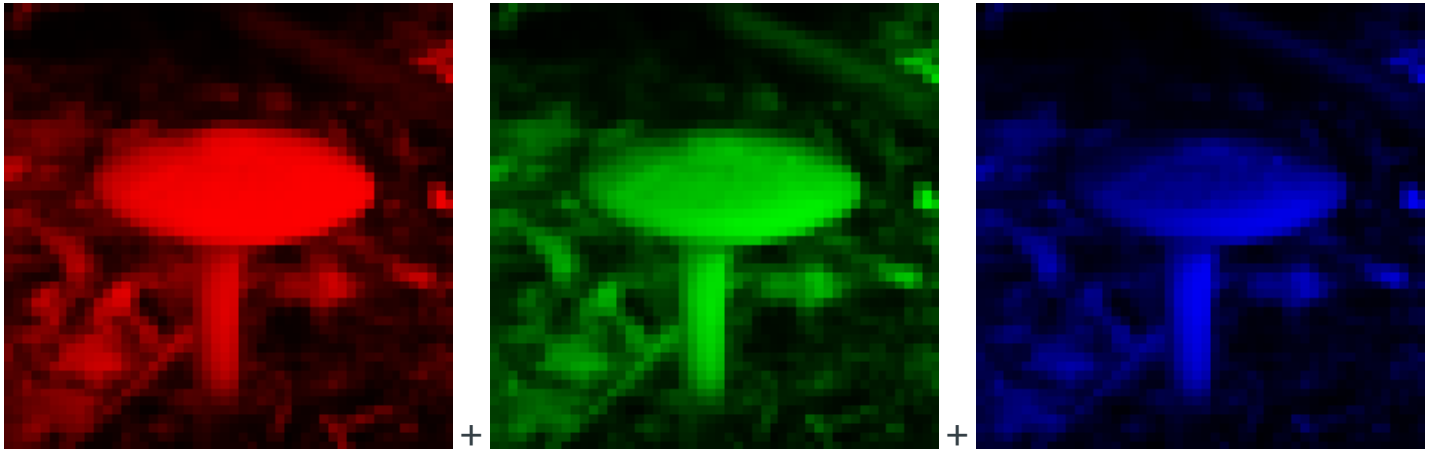
- image recognition
- speech processing
- natural language processing
- robotic control
- ... and many others.

How can we write a computer program that implements that?

The (human) brain is so good at interpreting visual information that the gap between raw data and its semantic interpretation is difficult to assess intuitively:

This is a mushroom.

This is a mushroom.



This is a mushroom.

```
array([[[0.03921569, 0.03529412, 0.02352941, 1.        ],
        [0.2509804 , 0.1882353 , 0.20392157, 1.        ],
        [0.4117647 , 0.34117648, 0.37254903, 1.        ],
        ...,
        [0.20392157, 0.23529412, 0.17254902, 1.        ],
        [0.16470589, 0.18039216, 0.12156863, 1.        ],
        [0.18039216, 0.18039216, 0.14117648, 1.        ]],

       [[0.1254902 , 0.11372549, 0.09411765, 1.        ],
        [0.2901961 , 0.2509804 , 0.24705882, 1.        ],
        [0.21176471, 0.2       , 0.20392157, 1.        ],
        ...,
        [0.1764706 , 0.24705882, 0.12156863, 1.        ],
        [0.10980392, 0.15686275, 0.07843138, 1.        ],
        [0.16470589, 0.20784314, 0.11764706, 1.        ]],

       [[0.14117648, 0.12941177, 0.10980392, 1.        ],
        [0.21176471, 0.1882353 , 0.16862746, 1.        ],
        [0.14117648, 0.13725491, 0.12941177, 1.        ],
        ...,
        [0.10980392, 0.15686275, 0.08627451, 1.        ],
        [0.0627451 , 0.08235294, 0.05098039, 1.        ],
        [0.14117648, 0.2       , 0.09803922, 1.        ]],

       ...,
```

This is a mushroom.

Extracting semantic information requires models of high complexity, which cannot be designed by hand.

However, one can write a program that learns the task of extracting semantic information.

Techniques used in practice consist of:

- defining a parametric model with high capacity,
- optimizing its parameters, by "making it work" on the training data.

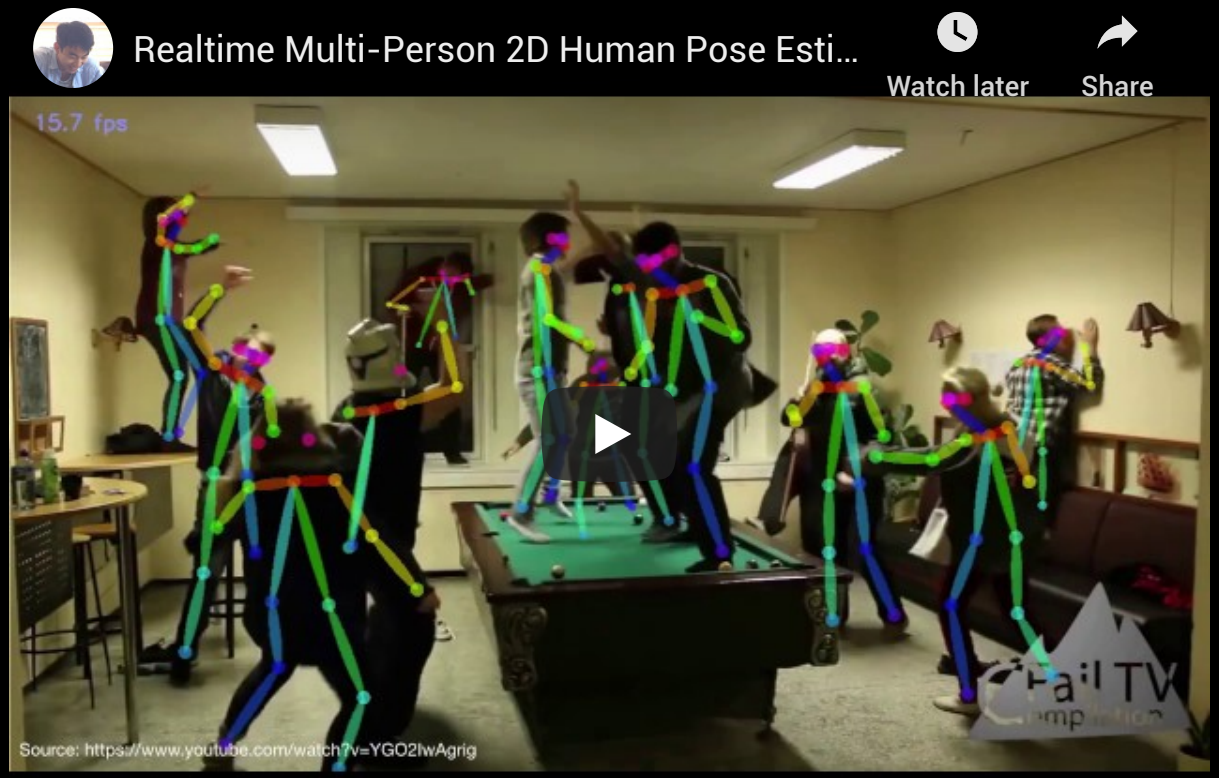
This is similar to biological systems for which the model (e.g., brain structure) is DNA-encoded, and parameters (e.g., synaptic weights) are tuned through experiences.

Deep learning encompasses software technologies to scale-up to billions of model parameters and as many training examples.

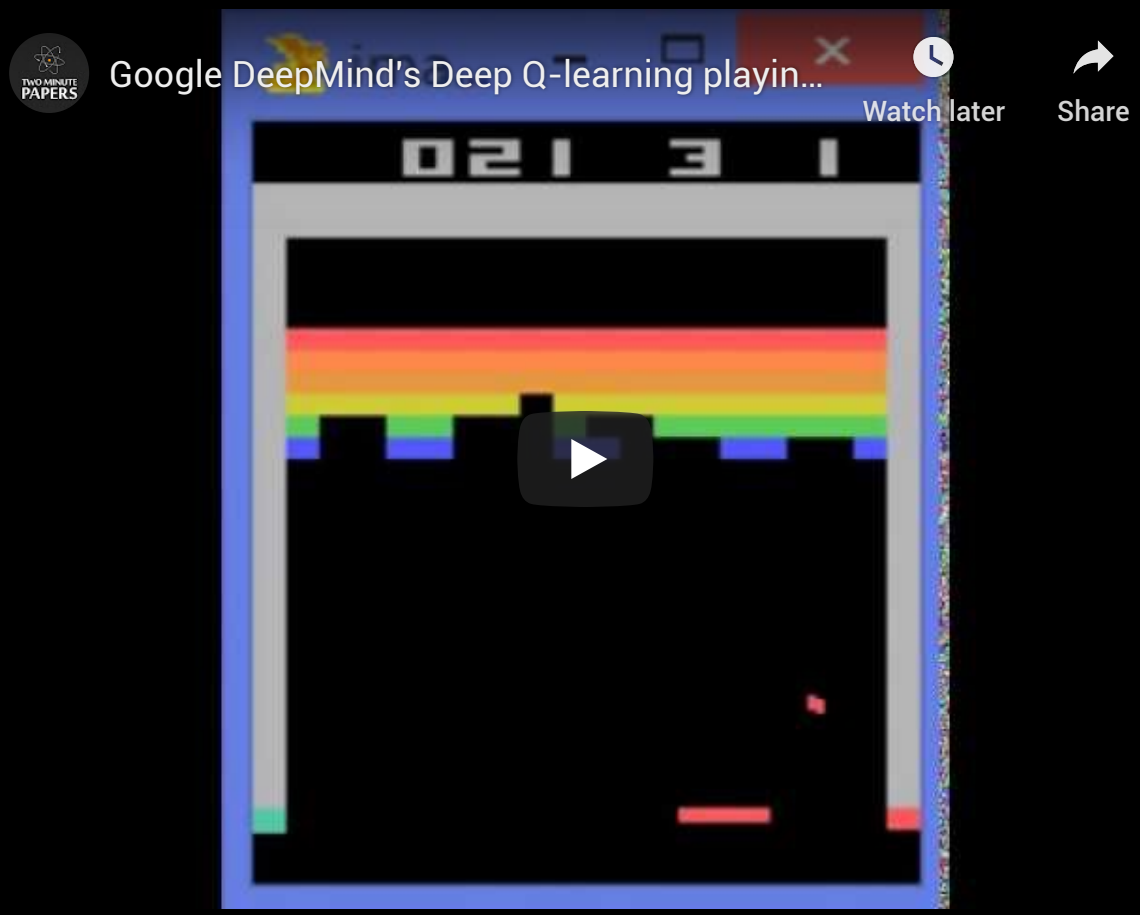
# Applications and success


Real-time object detection (Redmon and Farhadi, 2018)

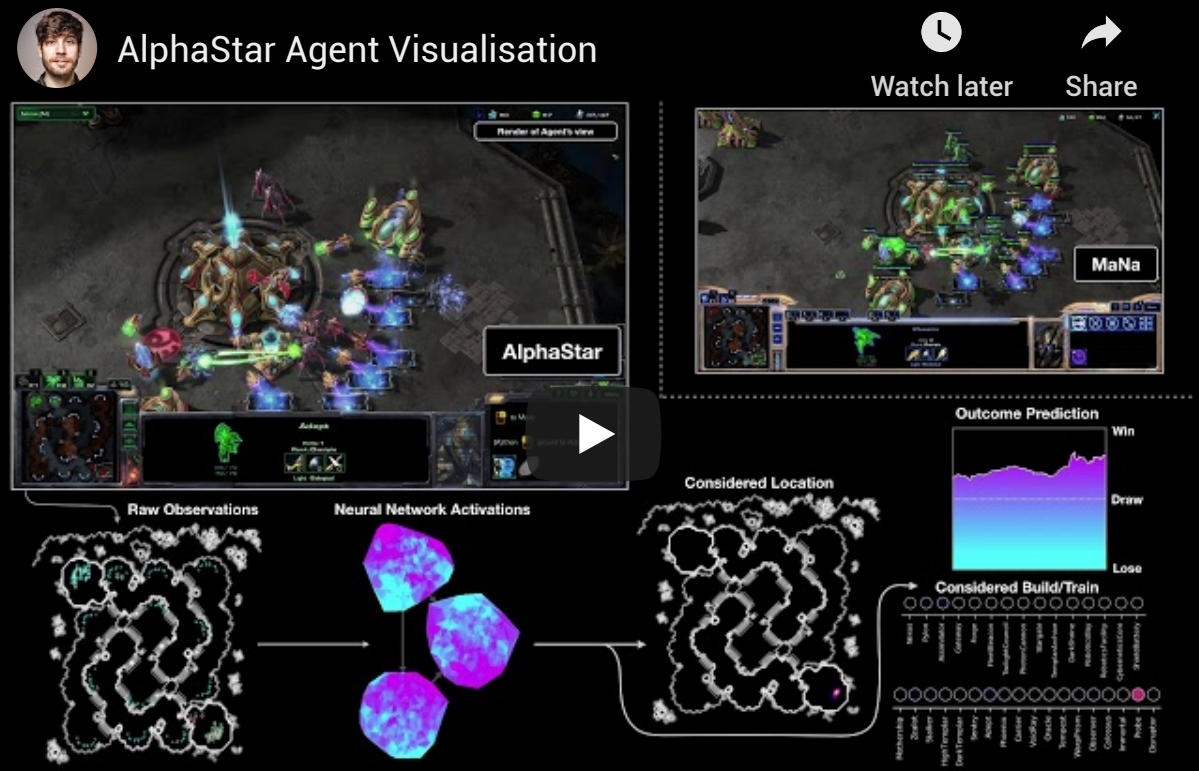

Segmentation (Hengshuang et al, 2017)




Pose estimation (Cao et al, 2017)




Reinforcement learning (Mnih et al, 2014)

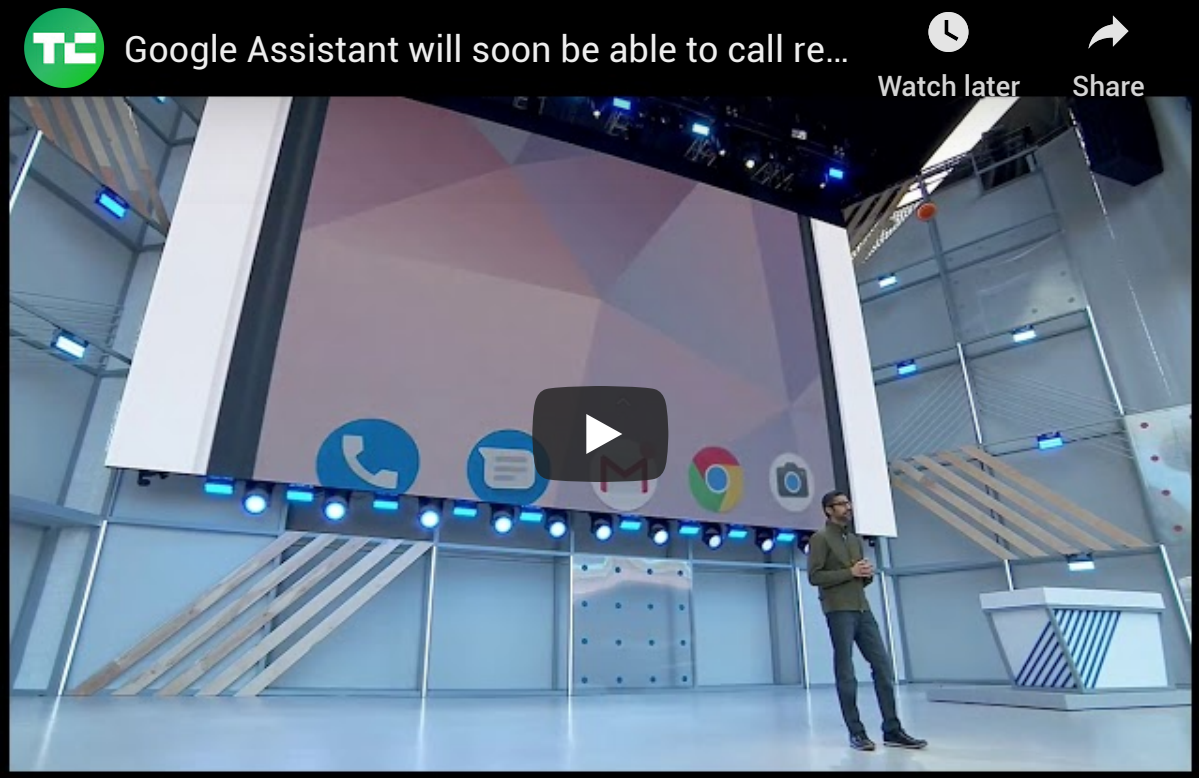



Strategy games (Deepmind, 2016-2018)


Autonomous cars (NVIDIA, 2016)


Speech recognition, translation and synthesis (Microsoft, 2012)


Auto-captioning (2015)




Speech synthesis and question answering (Google, 2018)

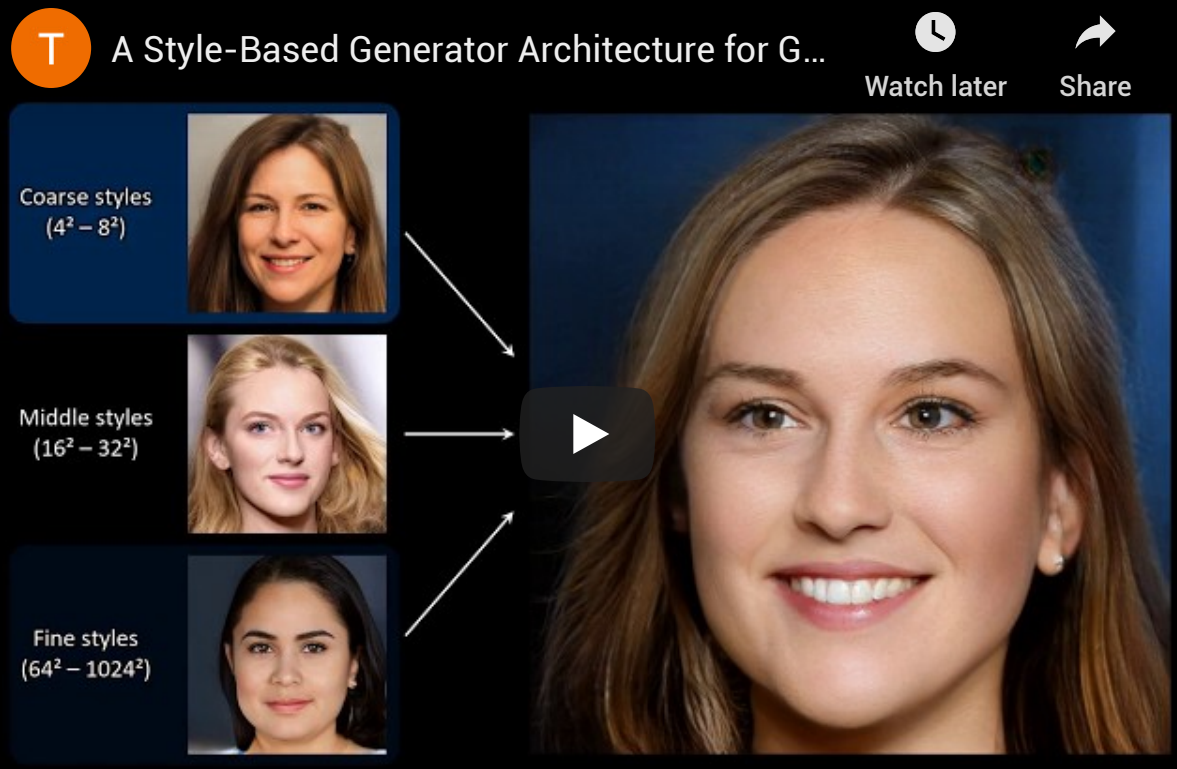
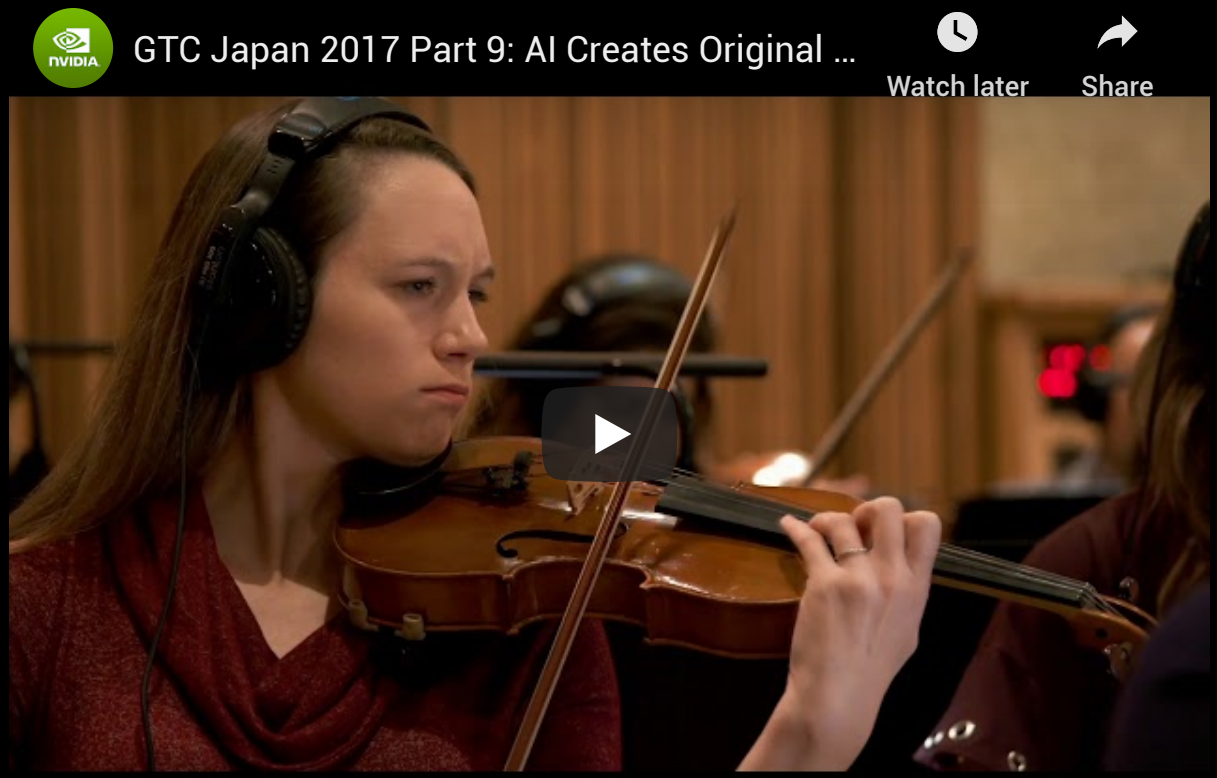



Image generation (Karras et al, 2018)

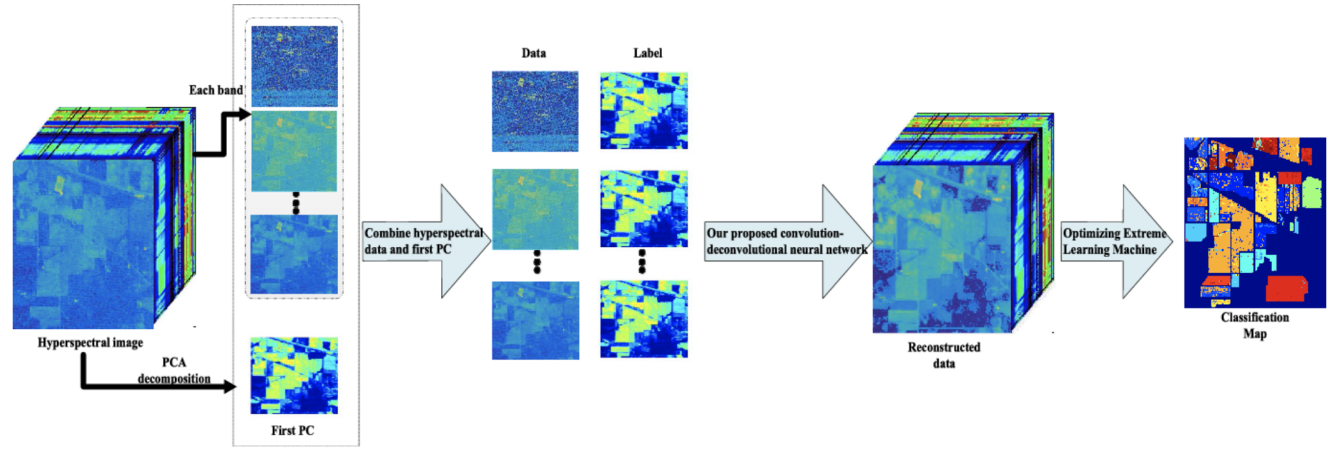



Music composition (NVIDIA, 2017)

# Deep Learning in Remote Sensing

# Landcover classification

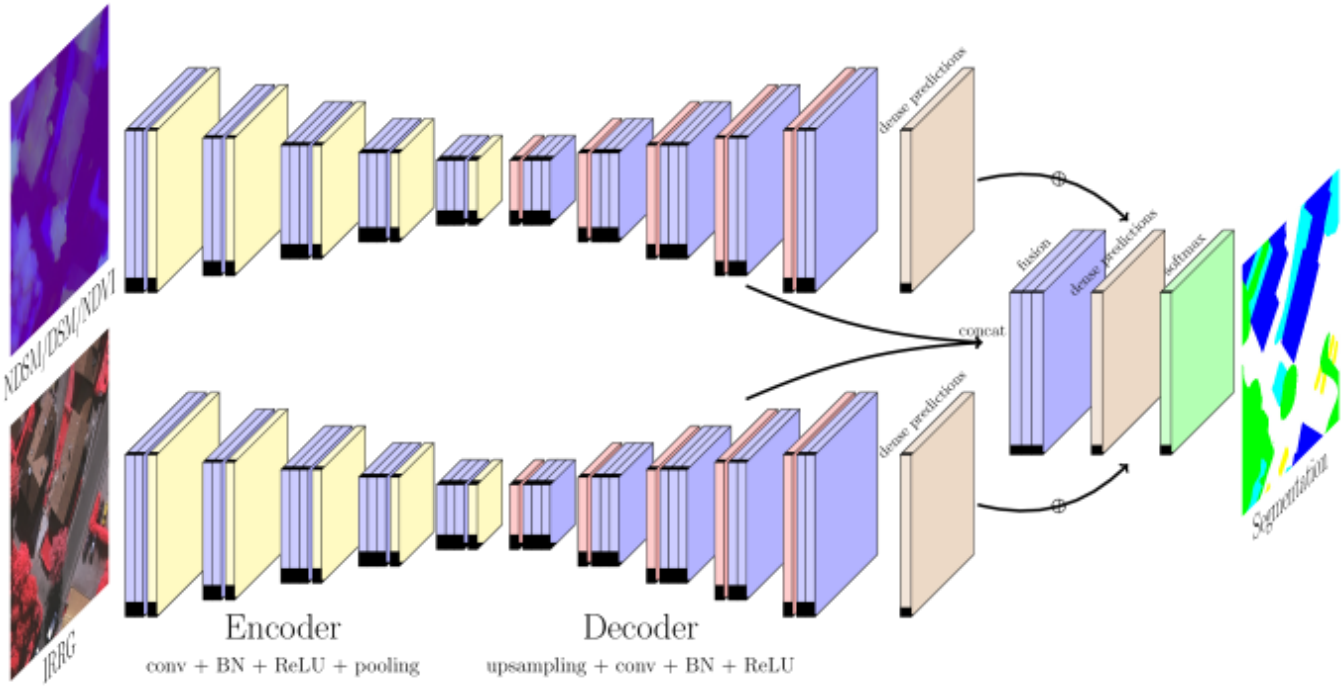
Hyperspectral image classification with fully convolutional neural networks




Credits: Jiaojiao Li et al., IEEE GRSL paper

# Semantic segmentation

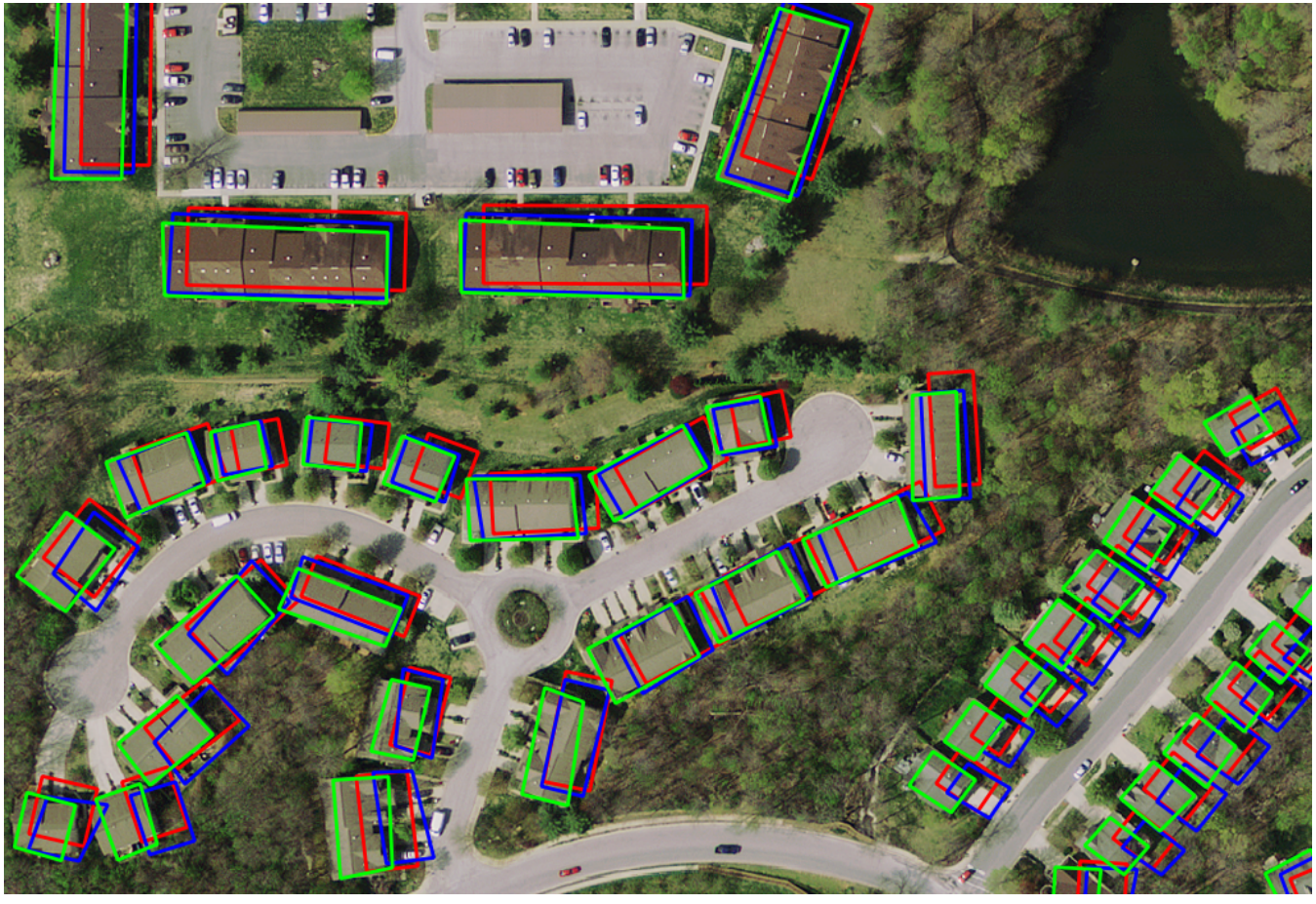
Fusion of modalities with different U-Net architectures




Credits: N. Audebert et al., Beyond RGB paper

# Registration

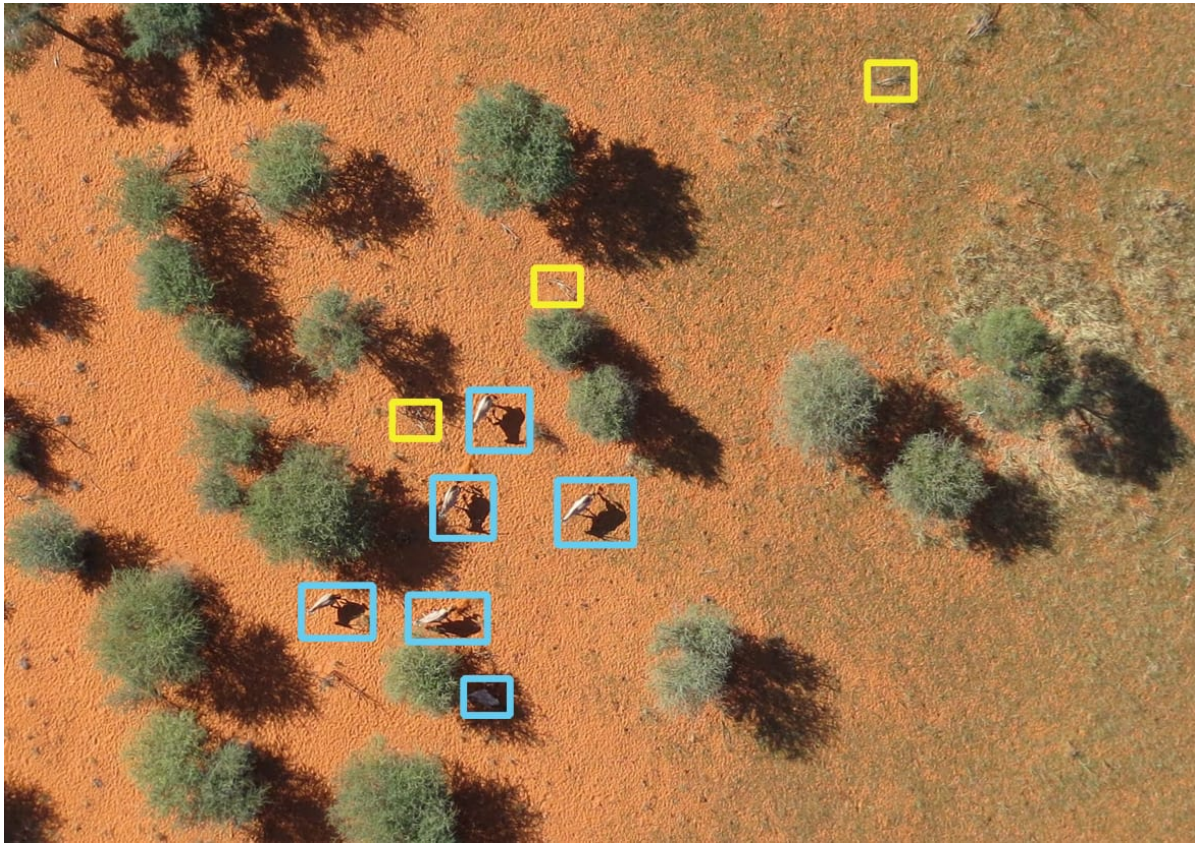
Registering RGB satellite images to cadaster maps



Credits: N. Girard et al., ACCV 2018 paper

# Wildlife Monitoring

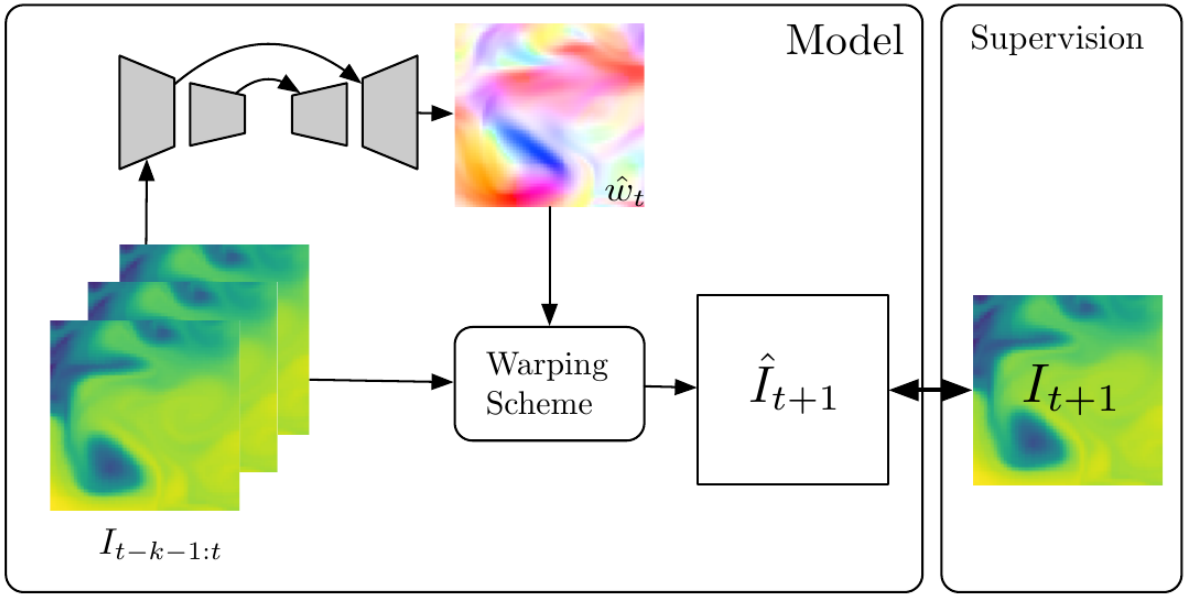
Detection of Antelopes in the Kuzikus parc, Namibia



Credits: D. Tuia et al., Drones Help Count African Wildlife

# Deep learning and Physics

Tracking the evolution of sea surface temperatures (SST)




Credits: Ayed et al., Learning Dynamical Systems from Partial Observations

# Deep learning and Physics

Simulation of ODE with neural networks

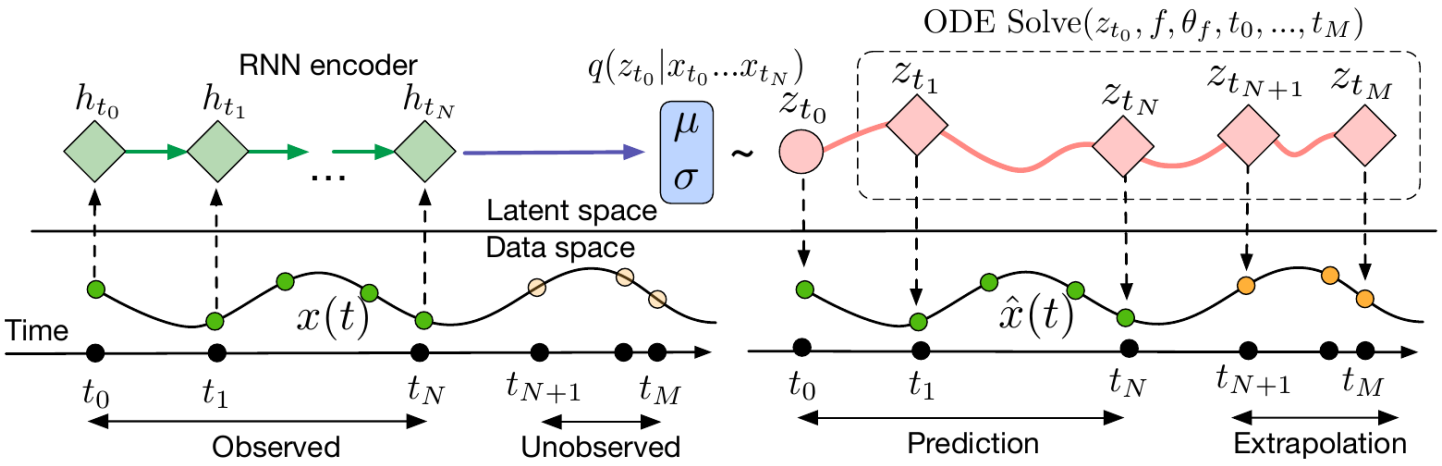
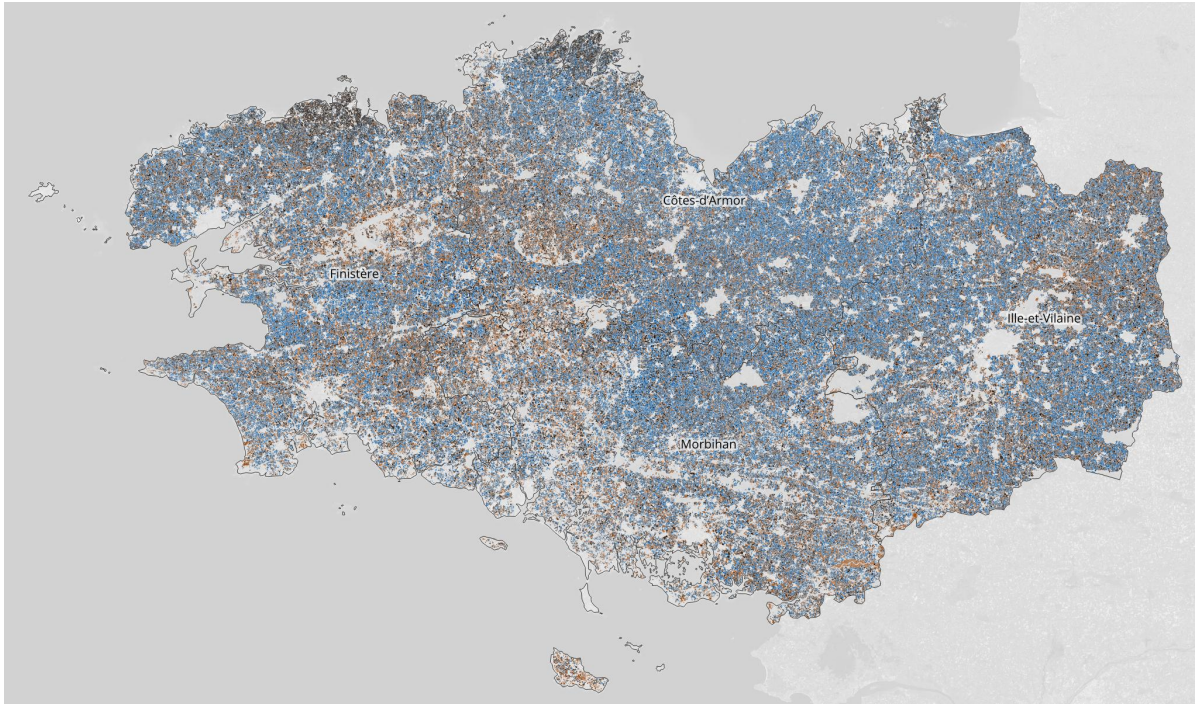



Figure 6: Computation graph of the latent ODE model.

# Mapping crops for Agriculture

Time series classification, BreizhCrops dataset




Credits: M. RuBwurm et al., Breizhcrops dataset (Github page)

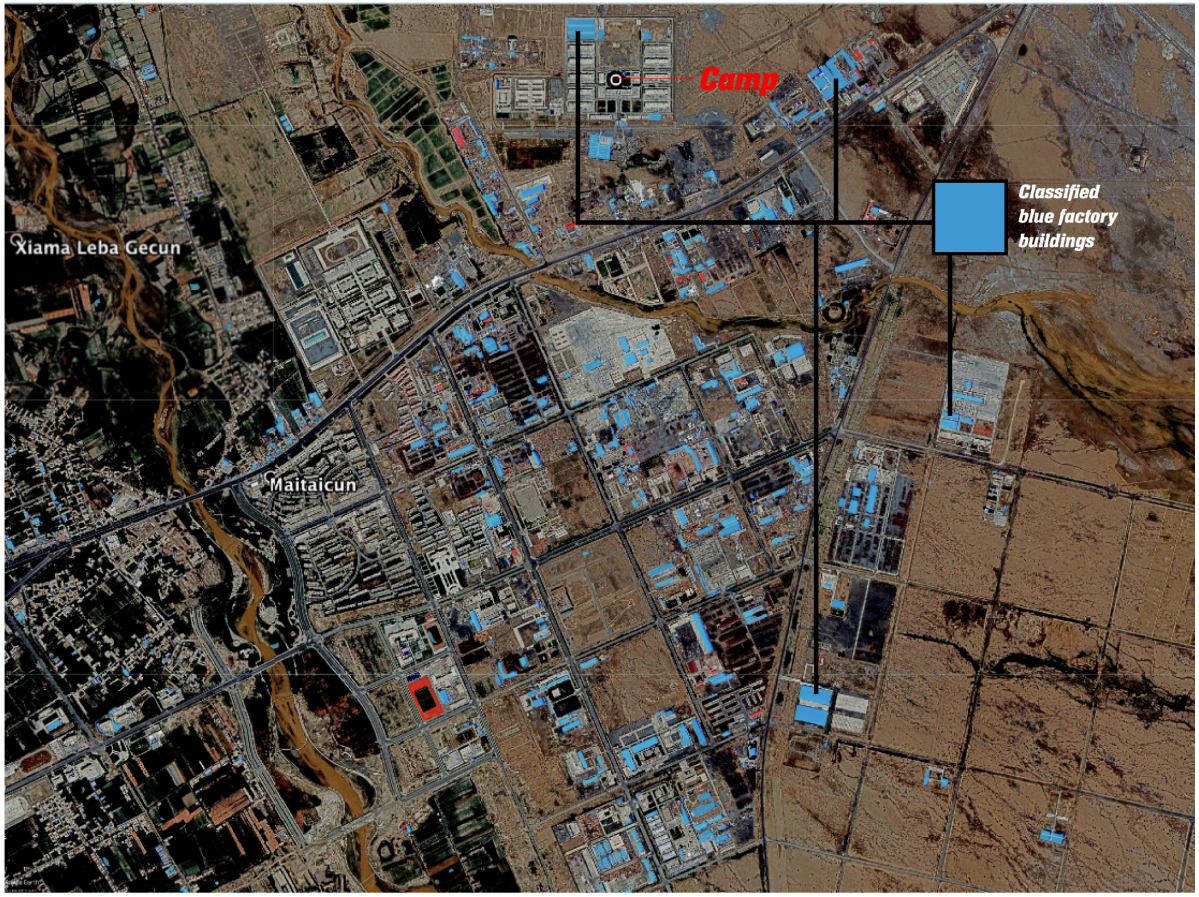
# Mapping after disaster

Damage assessment of the Beirut Harbour explosion


Credits: Xiaoxiang Zhu Lab, TU Münich, original tweet

# Mapping of changes

Assessing the devlopement of concentration camps in Xinjiang province




Credits: Techjournalist, original Medium blogspot Post

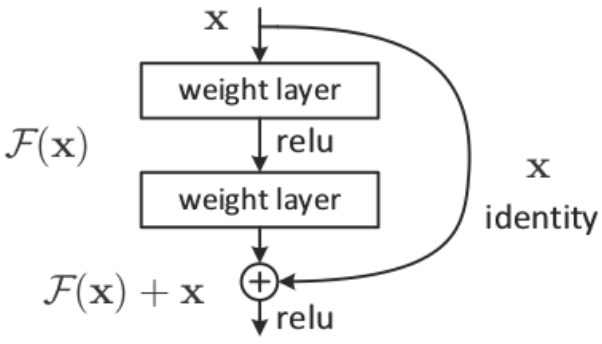
# Wrap-up

- Many different applications of deep learning in remote sensing
  - Ubiquitous in quasi all remote sensing fields
  - also a lot still to explore
- Many specific challenges
  - partially/weakly labelled data, noise in labels
  - acquisitions come in different nature (RGB, hyperpsectral, time-series, etc.). How to fuse those modalities ?
  - very-large scale amount of data (daily acquisitions, etc.): toward long-life learning
  - data come with physical knowledges about the underlying phenomenon. How to handle this ?
  - many more

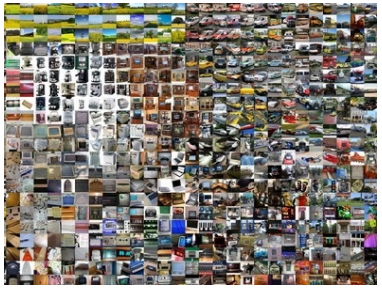
# Back to technique

# Why does it work now?

New algorithms




More data



Software

Caffe2 · Microsoft CNTK · TensorFlow · theano · Chainer · PYTORCH · dmlc mxnet · K

Faster compute engines

## Building on the shoulders of giants

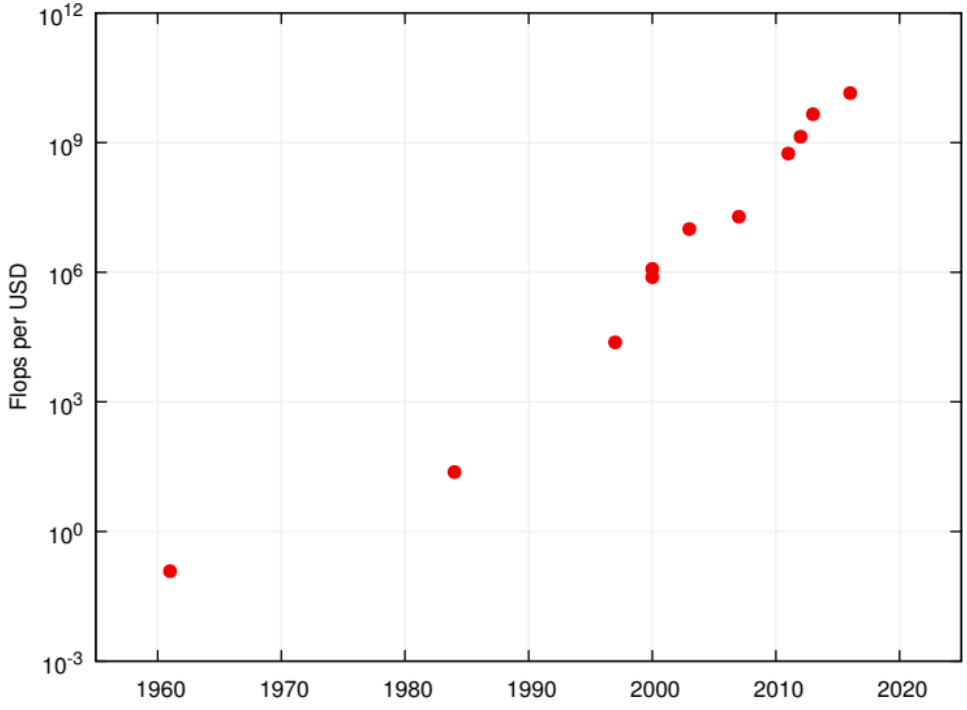
Five decades of research in machine learning provided

- a taxonomy of ML concepts (classification, generative models, clustering, kernels, linear embeddings, etc.),
- a sound statistical formalization (Bayesian estimation, PAC),
- a clear picture of fundamental issues (bias/variance dilemma, VC dimension, generalization bounds, etc.),
- a good understanding of optimization issues,
- efficient large-scale algorithms.

## Deep learning

From a practical perspective, deep learning

- lessens the need for a deep mathematical grasp,
- makes the design of large learning architectures a system/software development task,
- allows to leverage modern hardware (clusters of GPUs),
- does not plateau when using more data,
- makes large trained networks a commodity.




| | TFlops ($10^{12}$) | Price | GFlops per $ |
|---|---|---|---|
| Intel i7-6700K | 0.2 | $344 | 0.6 |
| AMD Radeon R-7 240 | 0.5 | $55 | 9.1 |
| NVIDIA GTX 750 Ti | 1.3 | $105 | 12.3 |
| AMD RX 480 | 5.2 | $239 | 21.6 |
| NVIDIA GTX 1080 | 8.9 | $699 | 12.7 |

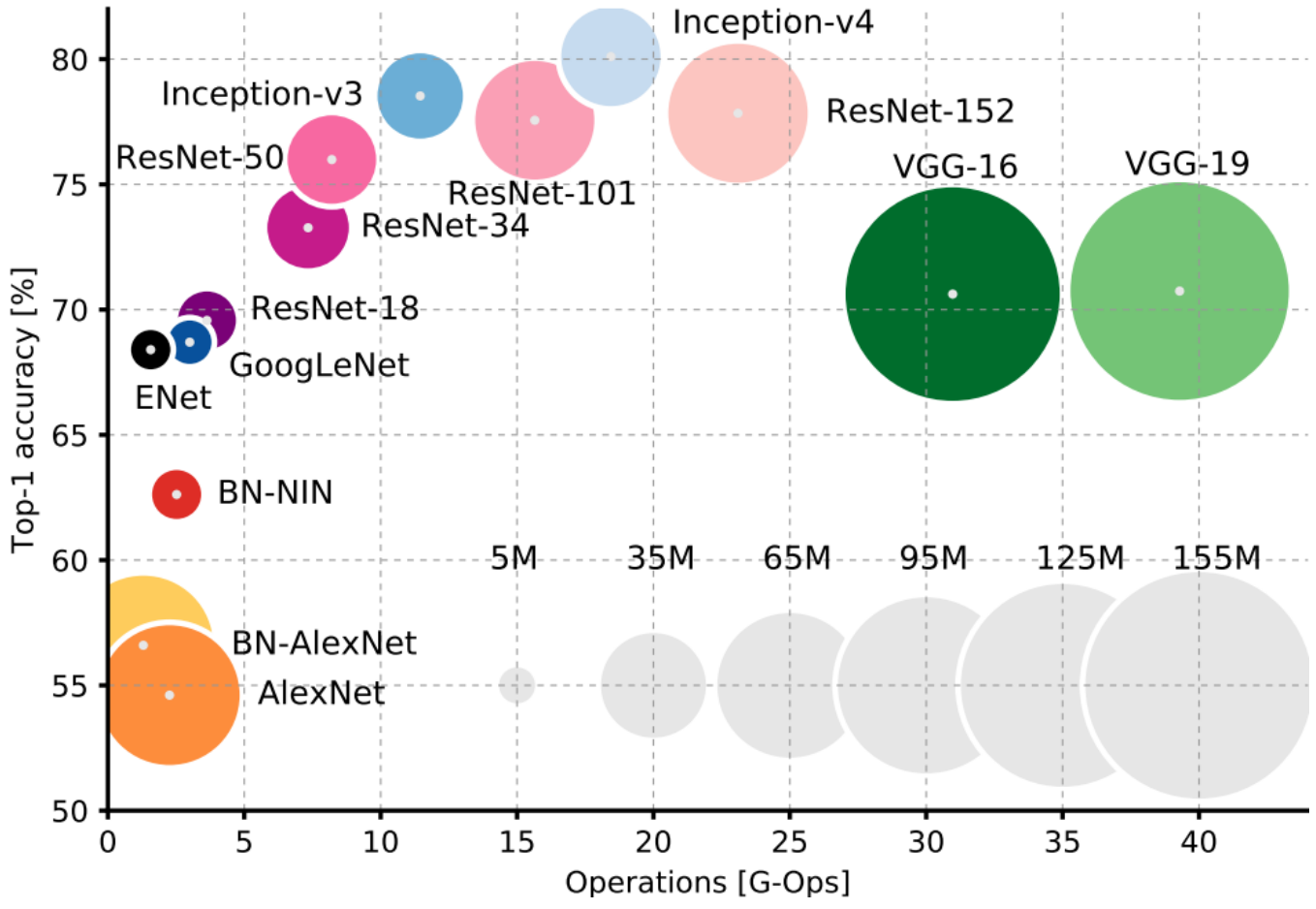



Image credits: Canziani et al, 2016, arXiv:1605.07678.

# Statistical learning

# Supervised learning

Consider an unknown joint probability distribution $P(X, Y)$.

Assume training data

$$(\mathbf{x}_i, y_i) \sim P(X, Y),$$

with $\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, ..., N$.

- In most cases,
  - $\mathbf{x}_i$ is a $p$-dimensional vector of features or descriptors,
  - $y_i$ is a scalar (e.g., a category or a real value).
- The training data is generated i.i.d.
- The training data can be of any finite size $N$.
- In general, we do not have any prior information about $P(X, Y)$.

## Inference

Supervised learning is usually concerned with the two following inference problems:

- Classification: Given $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \{1, ..., C\}$, for $i = 1, ..., N$, we want to estimate for any new $\mathbf{x}$,

$$\arg\max_y P(Y = y | X = \mathbf{x}).$$

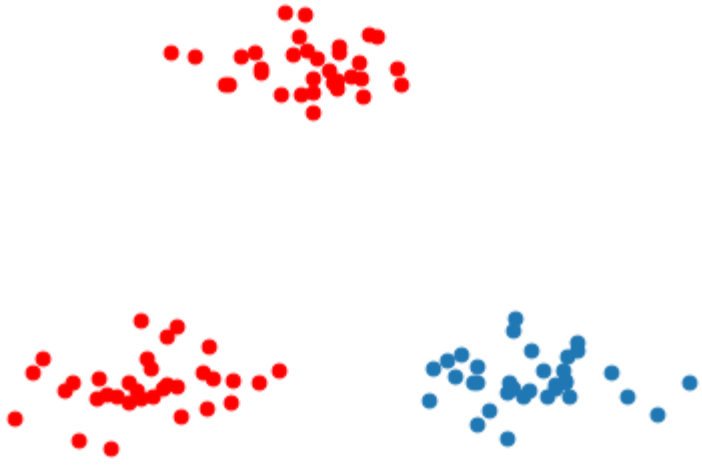
- Regression: Given $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \mathbb{R}$, for $i = 1, ..., N$, we want to estimate for any new $\mathbf{x}$,

$$\mathbb{E}\left[Y | X = \mathbf{x}\right].$$

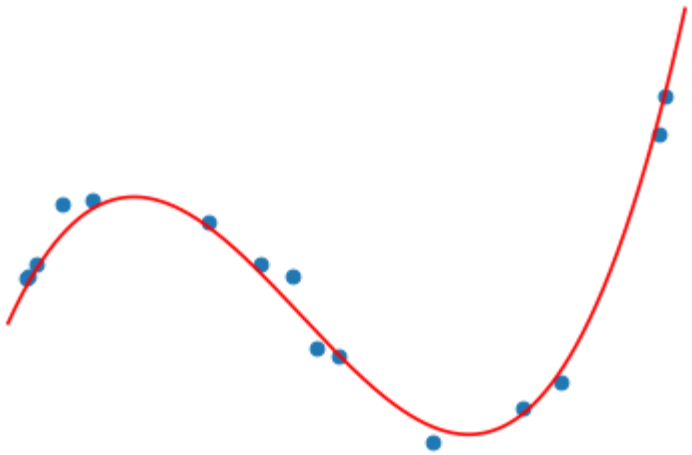
Or more generally, inference is concerned with the conditional estimation

$$P(Y = y|X = \mathbf{x})$$

for any new $(\mathbf{x}, y)$.



Classification consists in identifying
a decision boundary between objects of distinct classes.



Regression aims at estimating relationships among (usually continuous) variables.

# Empirical risk minimization

Consider a function $f : \mathcal{X} \to \mathcal{Y}$ produced by some learning algorithm. The predictions of this function can be evaluated through a loss

$$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R},$$

such that $\ell(y, f(\mathbf{x})) \geq 0$ measures how close the prediction $f(\mathbf{x})$ from $y$ is.

### Examples of loss functions

Classification: $\ell(y, f(\mathbf{x})) = \mathbf{1}_{y \neq f(\mathbf{x})}$

Regression: $\ell(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$

Let $\mathcal{F}$ denote the hypothesis space, i.e. the set of all functions $f$ than can be produced by the chosen learning algorithm.

We are looking for a function $f \in \mathcal{F}$ with a small expected risk (or generalization error)

$$R(f) = \mathbb{E}_{(\mathbf{x},y)\sim P(X,Y)} \left[\ell(y, f(\mathbf{x}))\right].$$

This means that for a given data generating distribution $P(X, Y)$ and for a given hypothesis space $\mathcal{F}$, the optimal model is

$$f_* = \arg\min_{f\in\mathcal{F}} R(f).$$

Unfortunately, since $P(X, Y)$ is unknown, the expected risk cannot be evaluated and the optimal model cannot be determined.

However, if we have i.i.d. training data $\mathbf{d} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, N\}$, we can compute an estimate, the empirical risk (or training error)

$$\hat{R}(f, \mathbf{d}) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{d}} \ell(y_i, f(\mathbf{x}_i)).$$

This estimate is unbiased and can be used for finding a good enough approximation of $f_*$. This results into the empirical risk minimization principle:

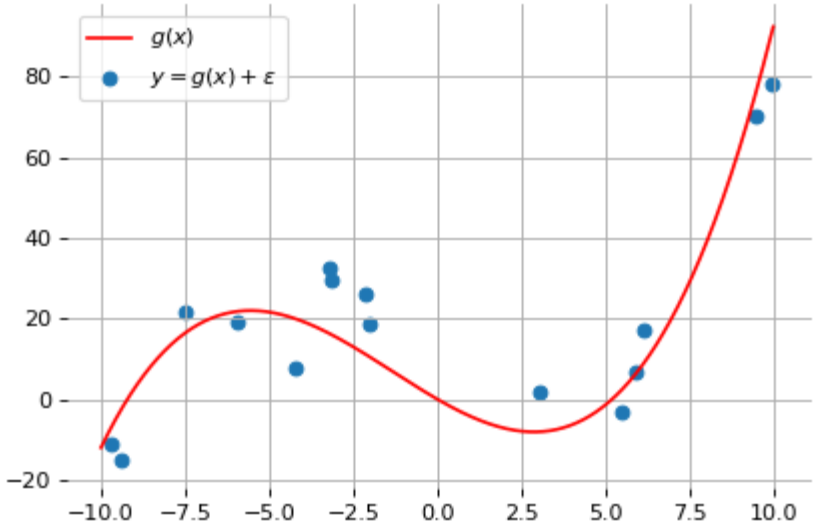
$$f_*^{\mathbf{d}} = \arg\min_{f \in \mathcal{F}} \hat{R}(f, \mathbf{d})$$

Most machine learning algorithms, including neural networks, implement empirical risk minimization.

Under regularity assumptions, empirical risk minimizers converge:

$$\lim_{N \to \infty} f_*^{\mathbf{d}} = f_*$$

# Polynomial regression




Consider the joint probability distribution $P(X, Y)$ induced by the data generating process

$$(x, y) \sim P(X, Y) \Leftrightarrow x \sim U[-10; 10], \epsilon \sim \mathcal{N}(0, \sigma^2), y = g(x) + \epsilon$$

where $x \in \mathbb{R}$, $y \in \mathbb{R}$ and $g$ is an unknown polynomial of degree 3.

Our goal is to find a function $f$ that makes good predictions on average over $P(X, Y)$.

Consider the hypothesis space $f \in \mathcal{F}$ of polynomials of degree 3 defined through their parameters $\mathbf{w} \in \mathbb{R}^4$ such that

$$\hat{y} \triangleq f(x; \mathbf{w}) = \sum_{d=0}^{3} w_d x^d$$

For this regression problem, we use the squared error loss

$$\ell(y, f(x; \mathbf{w})) = (y - f(x; \mathbf{w}))^2$$

to measure how wrong the predictions are.

Therefore, our goal is to find the best value $\mathbf{w}_*$ such

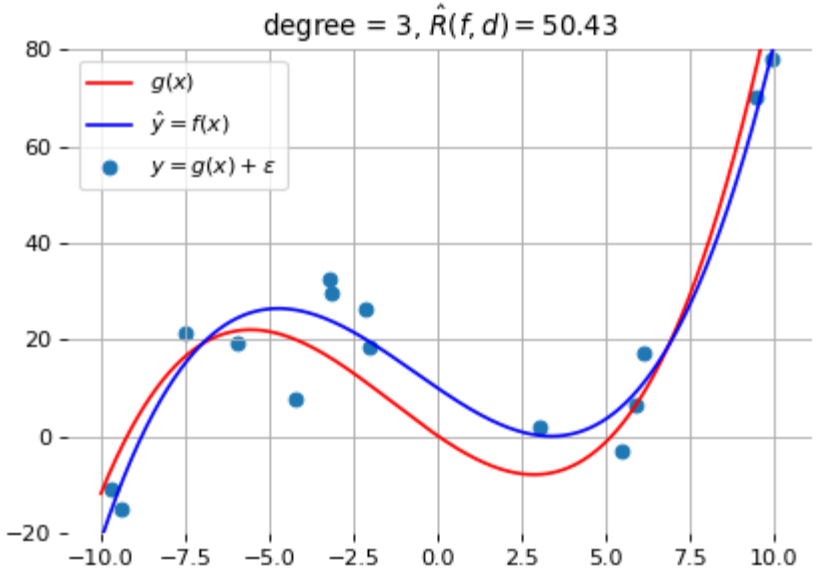
$$\begin{aligned}\mathbf{w}_* &= \arg\min_{\mathbf{w}} R(\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim P(X,Y)} \left[(y - f(x; \mathbf{w}))^2\right]\end{aligned}$$

Given a large enough training set $\mathbf{d} = \{(x_i, y_i) | i = 1, \ldots, N\}$, the empirical risk minimization principle tells us that a good estimate $\mathbf{w}_*^{\mathbf{d}}$ of $\mathbf{w}_*$ can be found by minimizing the empirical risk:

$$
\begin{aligned}
\mathbf{w}_*^{\mathbf{d}} &= \arg\min_{\mathbf{w}} \hat{R}(\mathbf{w}, \mathbf{d}) \\
&= \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(x_i, y_i) \in \mathbf{d}} (y_i - f(x_i; \mathbf{w}))^2 \\
&= \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(x_i, y_i) \in \mathbf{d}} (y_i - \sum_{d=0}^{3} w_d x_i^d)^2 \\
&= \arg\min_{\mathbf{w}} \frac{1}{N} \left\| \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{pmatrix}}_{\mathbf{y}} - \underbrace{\begin{pmatrix} x_1^0 \ldots x_1^3 \\ x_2^0 \ldots x_2^3 \\ \ldots \\ x_N^0 \ldots x_N^3 \end{pmatrix}}_{\mathbf{X}} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} \right\|^2
\end{aligned}
$$

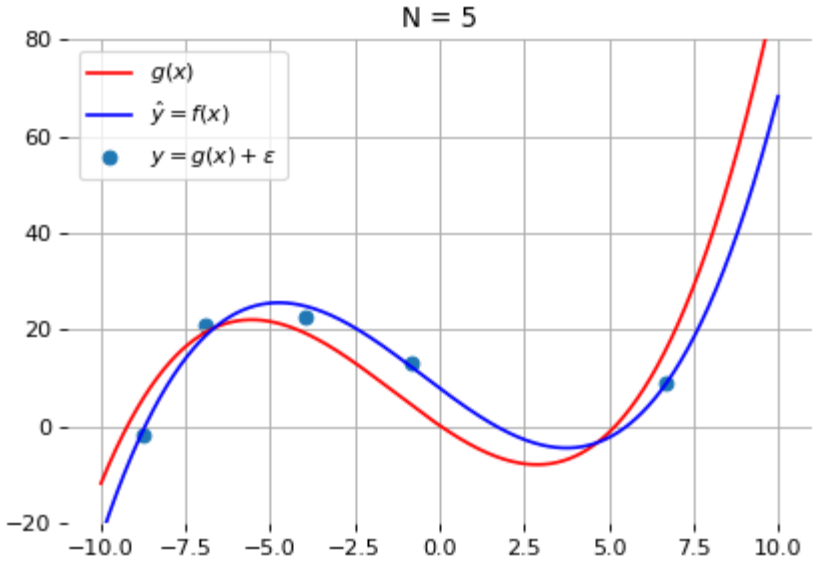
This is ordinary least squares regression, for which the solution is known analytically:

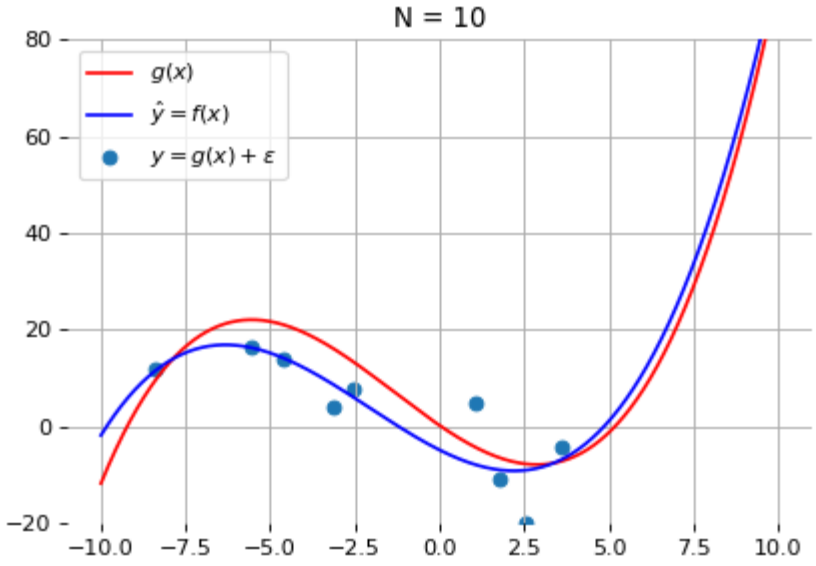
$$\mathbf{w}_*^{\mathbf{d}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

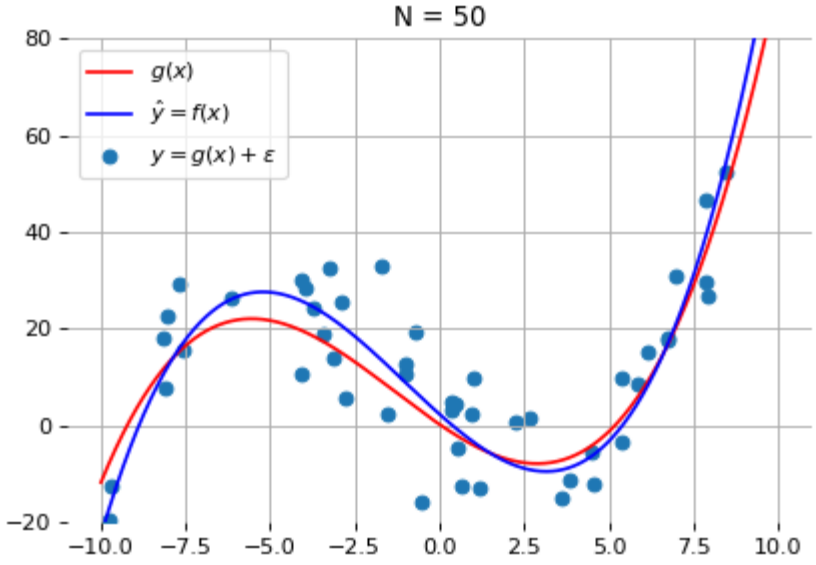


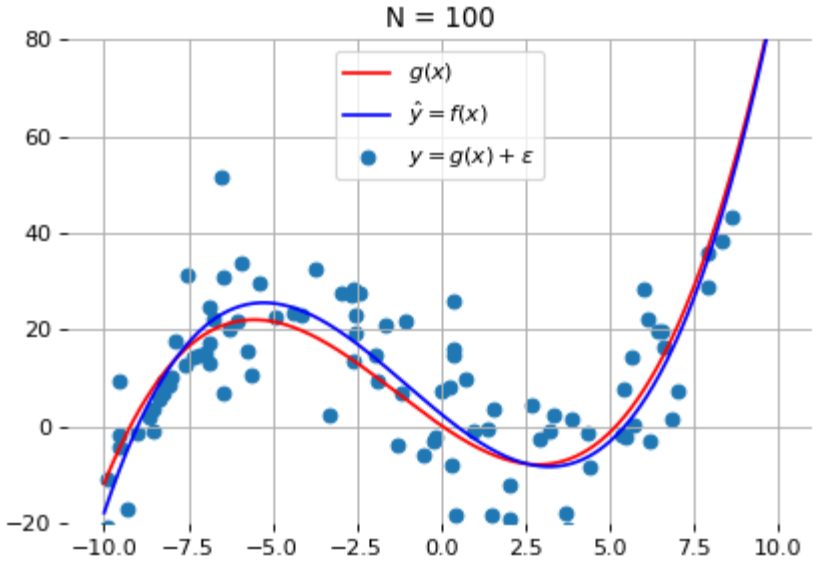
The expected risk minimizer $\mathbf{w}_*$ within our hypothesis space is $g$ itself.

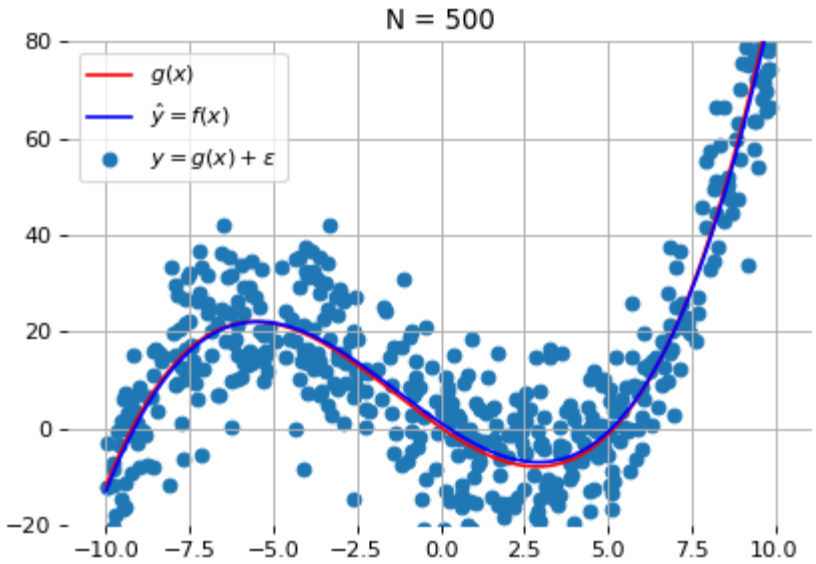
Therefore, on this toy problem, we can verify that $f(x; \mathbf{w}_*^{\mathbf{d}}) \to f(x; \mathbf{w}_*) = g(x)$ as $N \to \infty$.


N = 5
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$


N = 10
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

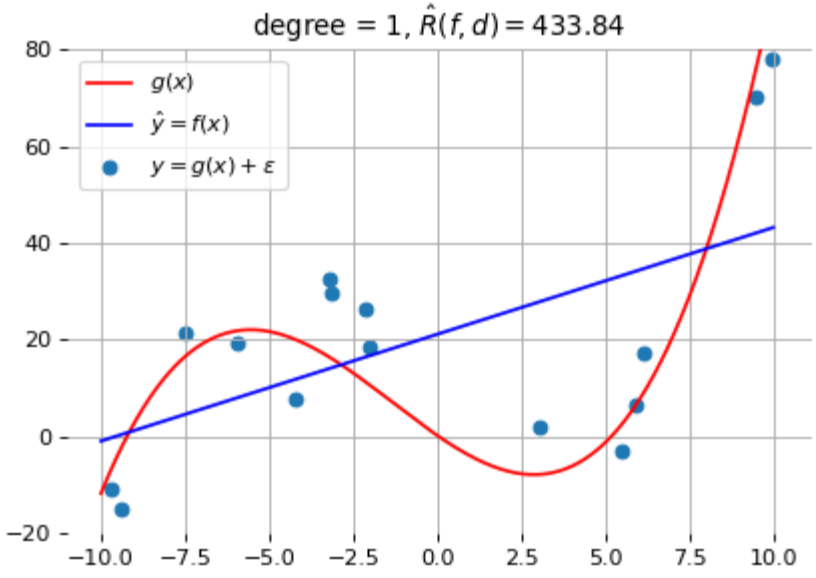

N = 50
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


N = 100
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

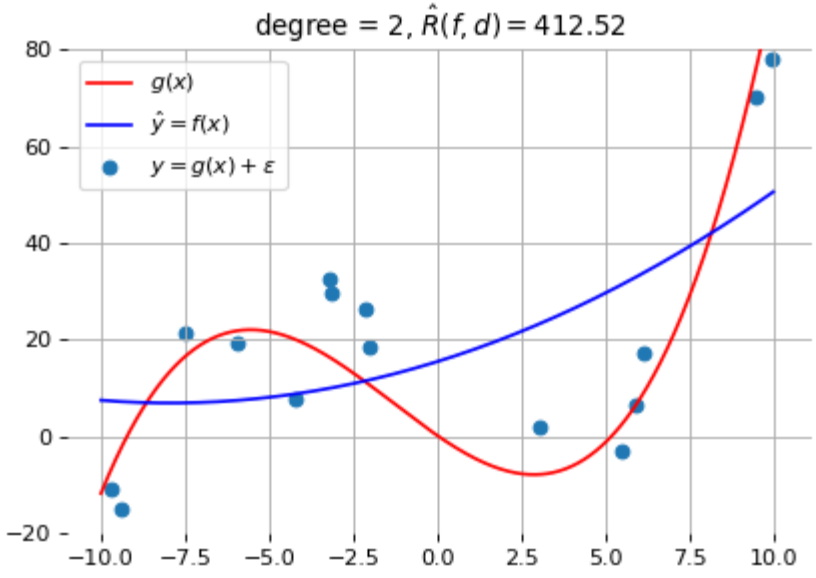

N = 500
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

# Under-fitting and over-fitting

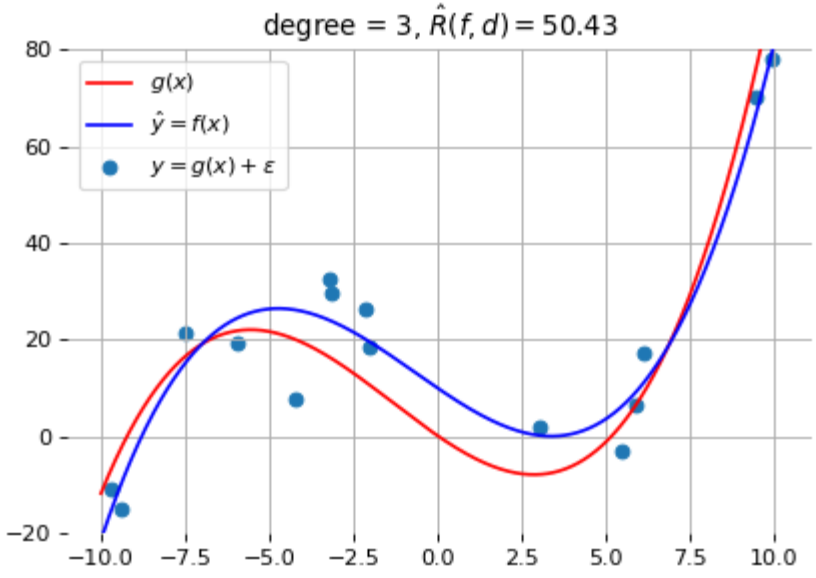
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions $f$ are either too "simple" or too "complex" with respect to the true data generating process?



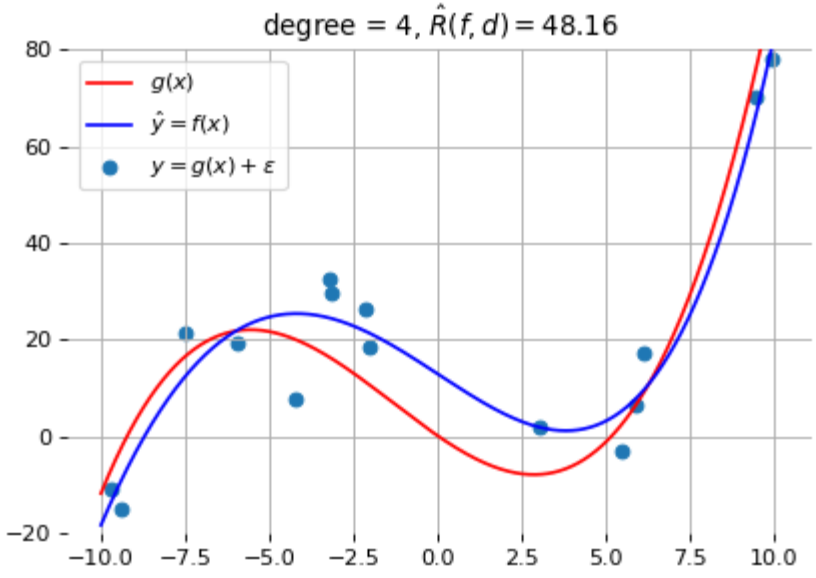

$\mathcal{F}$ = polynomials of degree 1



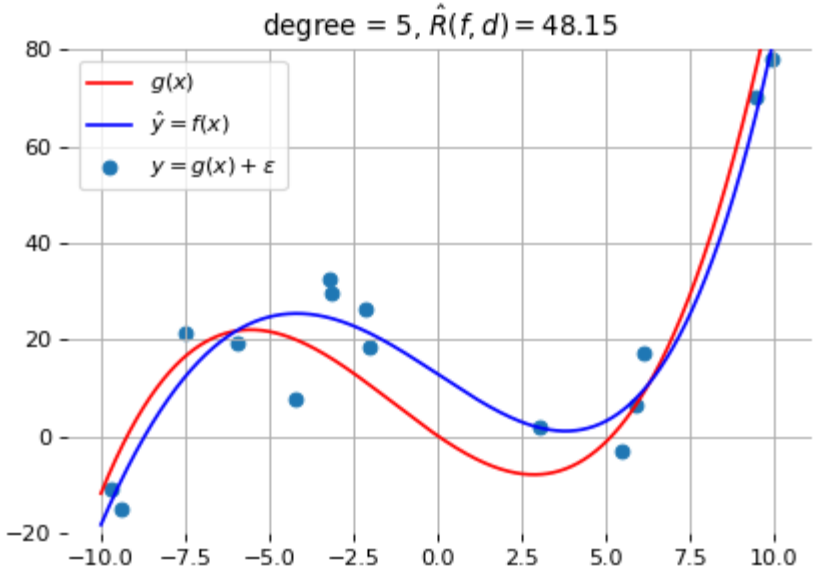

$\mathcal{F}$ = polynomials of degree 2



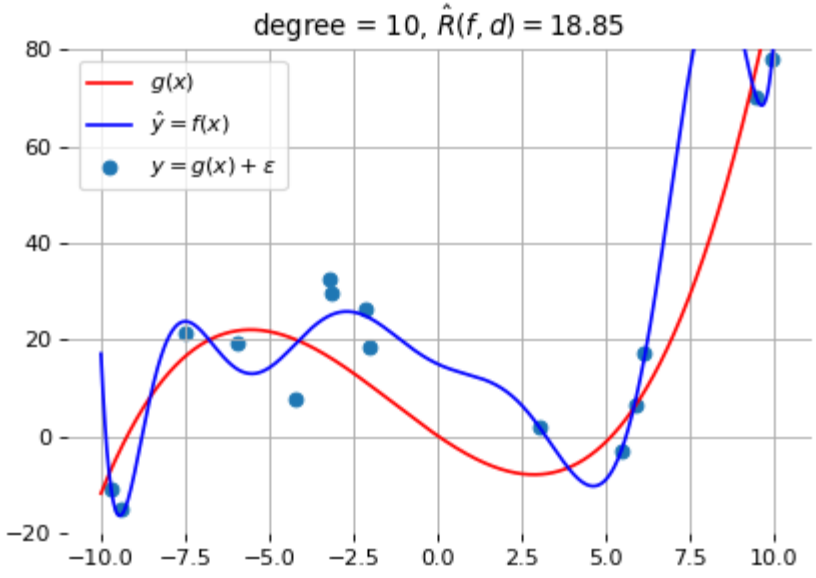

$\mathcal{F}$ = polynomials of degree 3



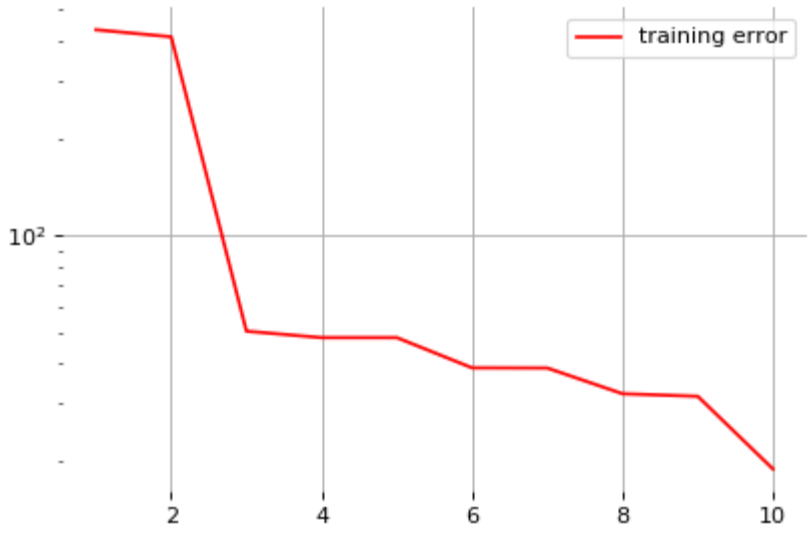

$\mathcal{F}$ = polynomials of degree 4




$\mathcal{F}$ = polynomials of degree 5




$\mathcal{F}$ = polynomials of degree 10




Degree $d$ of the polynomial VS. error.

Let $\mathcal{Y}^{\mathcal{X}}$ be the set of all functions $f : \mathcal{X} \to \mathcal{Y}$.

We define the Bayes risk as the minimal expected risk over all possible functions,

$$R_B = \min_{f \in \mathcal{Y}^{\mathcal{X}}} R(f),$$

and call Bayes model the model $f_B$ that achieves this minimum.

No model $f$ can perform better than $f_B$.

The capacity of an hypothesis space induced by a learning algorithm intuitively represents the ability to find a good model $f \in \mathcal{F}$ for any function, regardless of its complexity.

In practice, capacity can be controlled through hyper-parameters of the learning algorithm. For example:

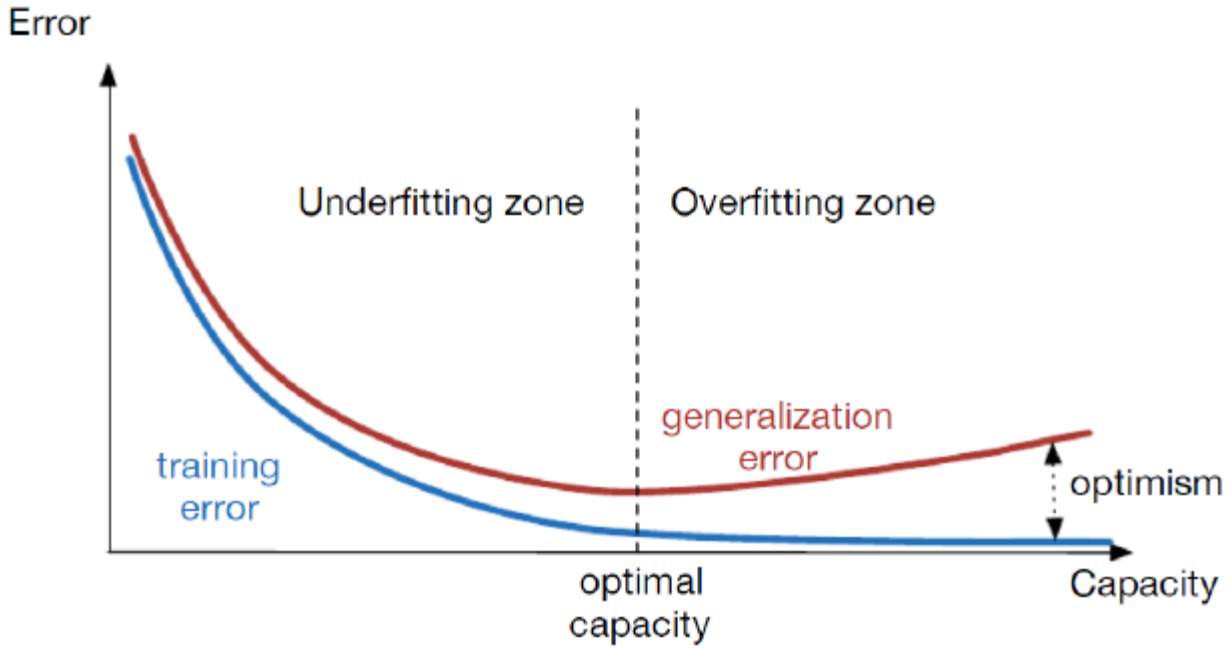
- The degree of the family of polynomials;
- The number of layers in a neural network;
- The number of training iterations;
- Regularization terms.

- If the capacity of $\mathcal{F}$ is too low, then $f_B \notin \mathcal{F}$ and $R(f) - R_B$ is large for any $f \in \mathcal{F}$, including $f_*$ and $f_*^{\mathbf{d}}$. Such models $f$ are said to underfit the data.
- If the capacity of $\mathcal{F}$ is too high, then $f_B \in \mathcal{F}$ or $R(f_*) - R_B$ is small. However, because of the high capacity of the hypothesis space, the empirical risk minimizer $f_*^{\mathbf{d}}$ could fit the training data arbitrarily well such that

$$R(f_*^{\mathbf{d}}) \geq R_B \geq \hat{R}(f_*^{\mathbf{d}}, \mathbf{d}) \geq 0.$$

  In this situation, $f_*^{\mathbf{d}}$ becomes too specialized with respect to the true data generating process and a large reduction of the empirical risk (often) comes at the price of an increase of the expected risk of the empirical risk minimizer $R(f_*^{\mathbf{d}})$. In this situation, $f_*^{\mathbf{d}}$ is said to overfit the data.

Therefore, our goal is to adjust the capacity of the hypothesis space such that the expected risk of the empirical risk minimizer gets as low as possible.



When overfitting,

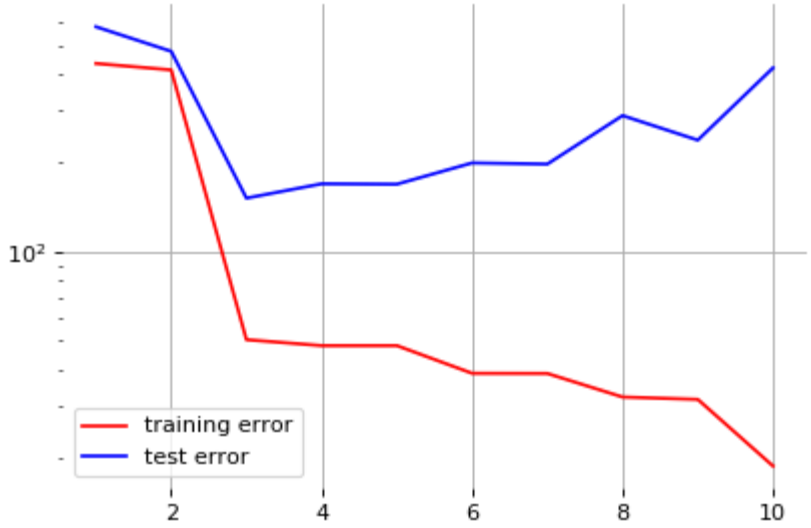
$$R(f_*^{\mathbf{d}}) \geq R_B \geq \hat{R}(f_*^{\mathbf{d}}, \mathbf{d}) \geq 0.$$

This indicates that the empirical risk $\hat{R}(f_*^{\mathbf{d}}, \mathbf{d})$ is a poor estimator of the expected risk $R(f_*^{\mathbf{d}})$.

Nevertheless, an unbiased estimate of the expected risk can be obtained by evaluating $f_*^{\mathbf{d}}$ on data $\mathbf{d}_{\text{test}}$ independent from the training samples $\mathbf{d}$:

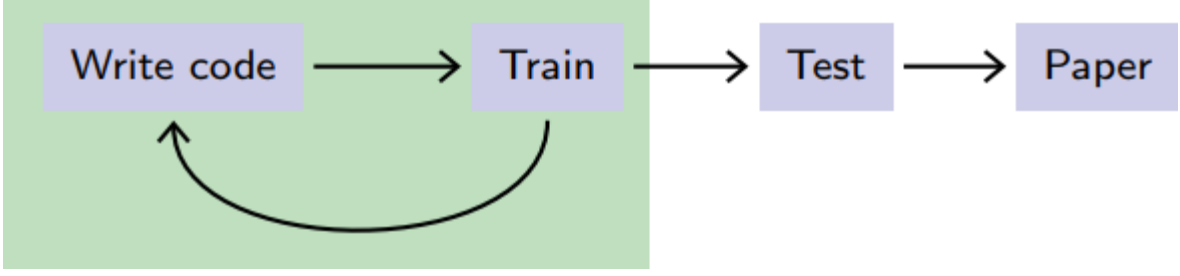
$$\hat{R}(f_*^{\mathbf{d}}, \mathbf{d}_{\text{test}}) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{d}_{\text{test}}} \ell(y_i, f_*^{\mathbf{d}}(\mathbf{x}_i))$$

This test error estimate can be used to evaluate the actual performance of the model. However, it should not be used, at the same time, for model selection.

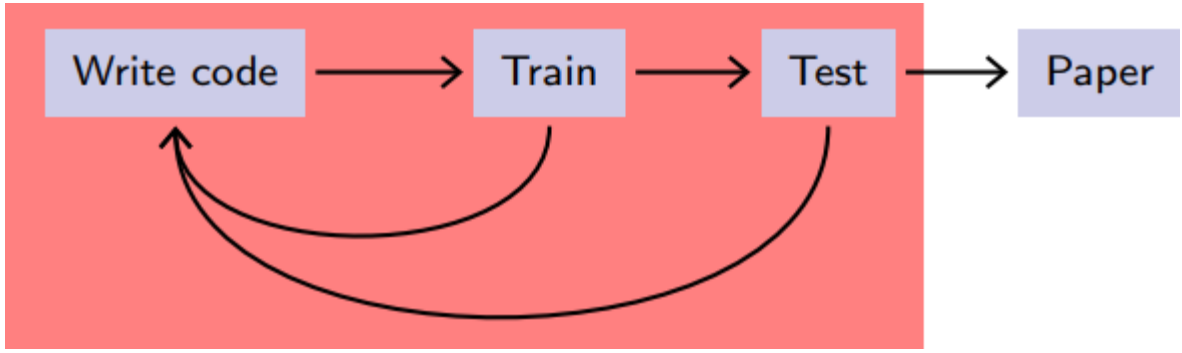



Degree $d$ of the polynomial VS. error.

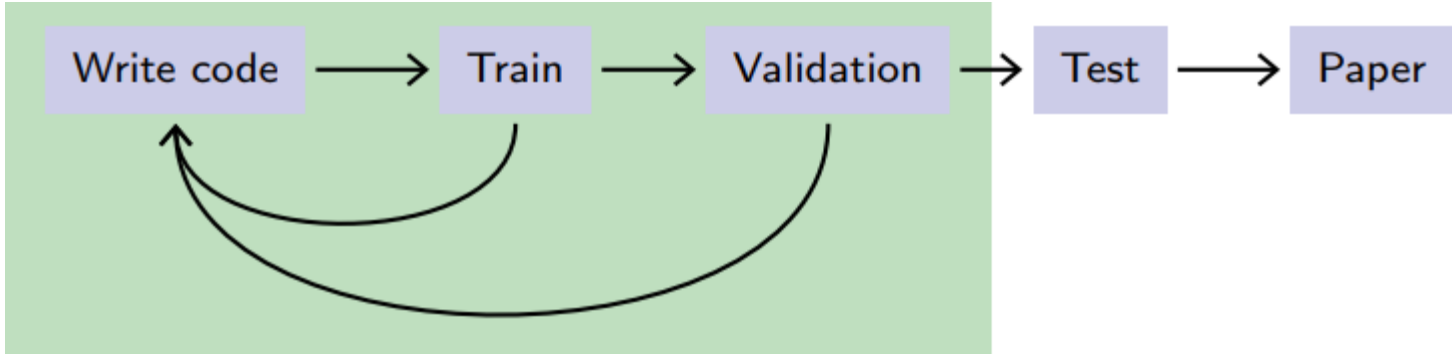
## (Proper) evaluation protocol




There may be over-fitting, but it does not bias the final performance evaluation.




This should be avoided at all costs!




Instead, keep a separate validation set for tuning the hyper-parameters.

The end.

# References / Credits

Elements of this course can be found in

- Vapnik, V. (1992). Principles of risk minimization for learning theory. In Advances in neural information processing systems (pp. 831-838).
- Deep Learning Book, Goodfellow, Bengio and Courville (free web access)
- Deep learning course from François Fleuret (EPFL) .
- Deep learning course from Gilles Louppe (U. Liège).
- cours sur les réseaux neuronaux Hugo Larochelle (U. Sherbrook and Google Brain Montréal).