Machine learning and physical (Earth system) modelling

julien.brajard@nersc.no

June 2021

NERSC

slides+notebook:*https://github.com/nansencenter/ml-crashcourse*

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
Deep Learning.
MIT Press, 2016.
http://www.deeplearningbook.org.

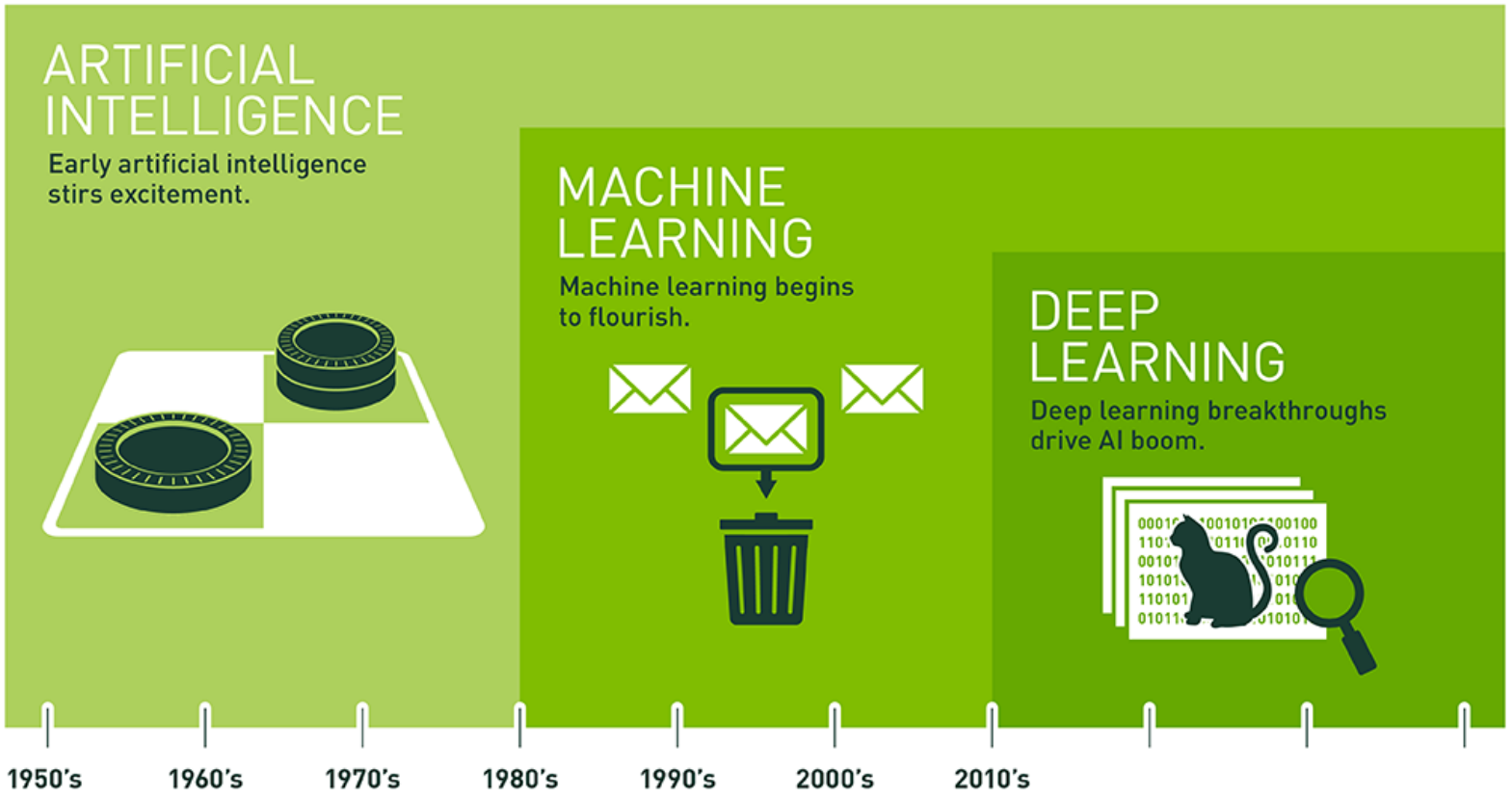
Jake VanderPlas.
Python Data Science Handbook: Essential Tools for Working with Data.
O'Reilly Media, Inc., 1st edition, 2016.

Table of contents i

Table of contents ii

Introduction

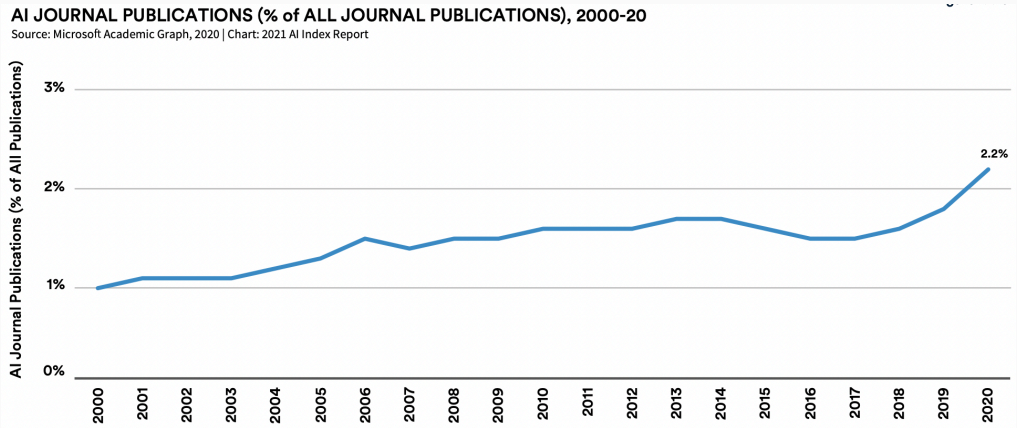
Scope of the lecture: Machine Learning

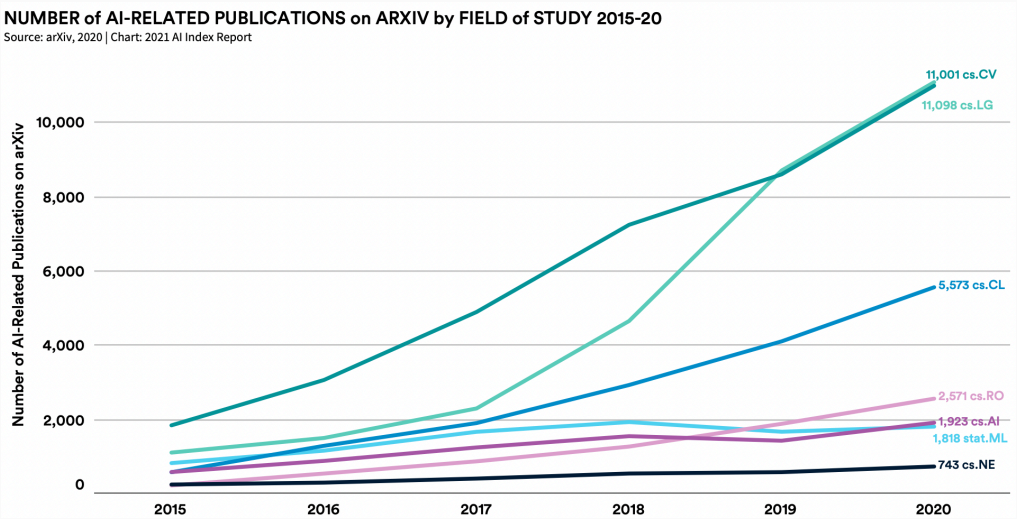


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: NVidia

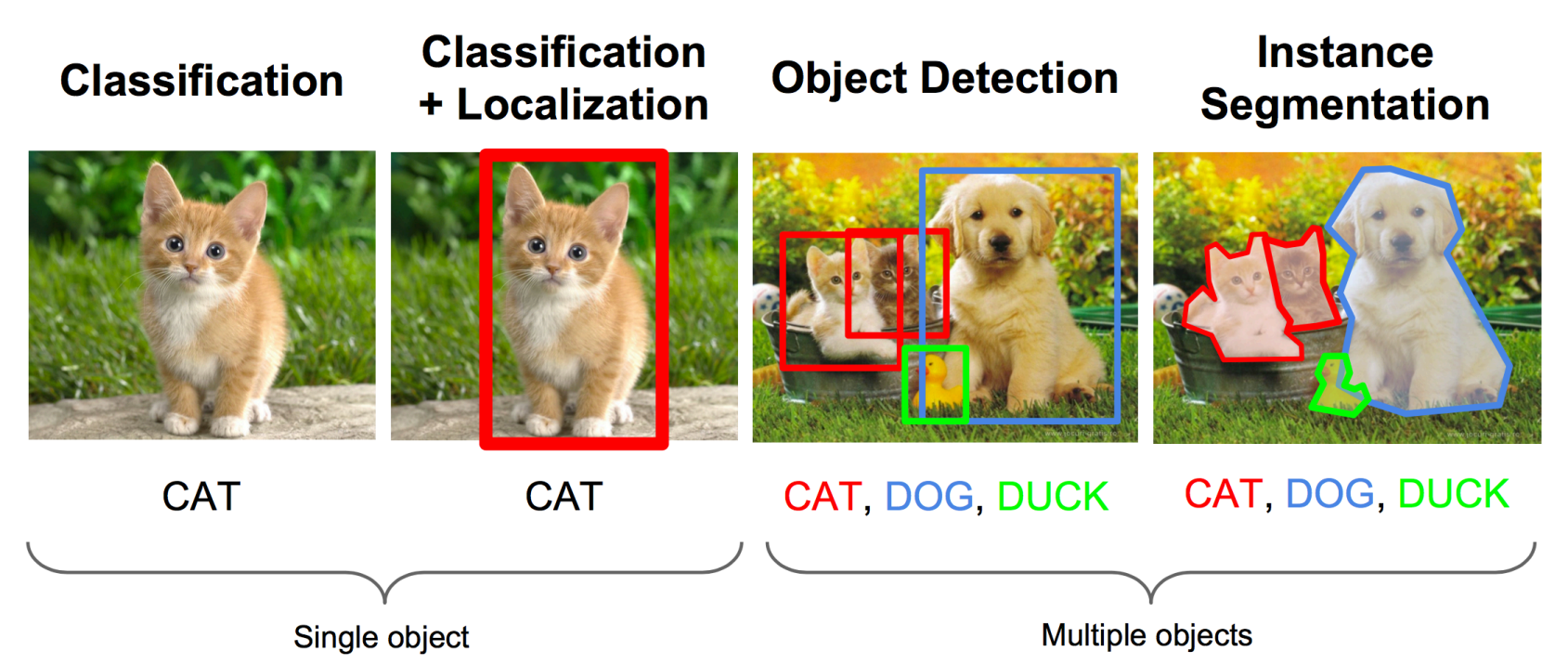
A (very) active field





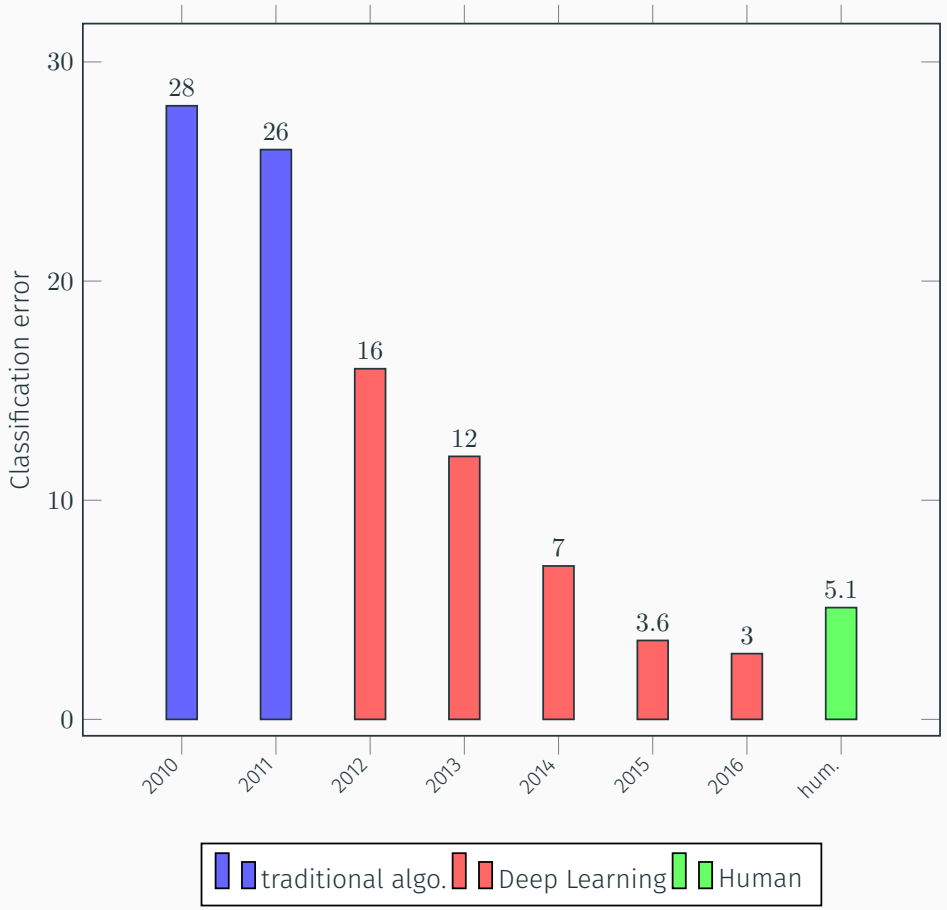
Zhang et al., "The AI Index 2021 Annual Report"

Example 1: Computer Vision



Li, Karpathy and Johnson, 2016, Stanford CS231n course

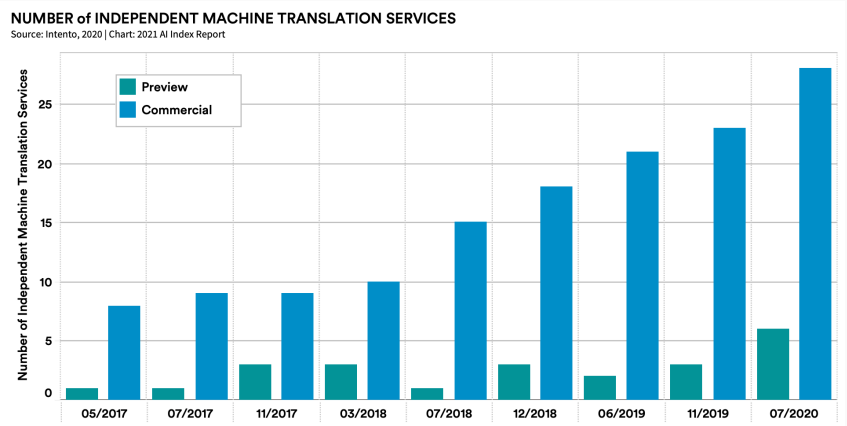
Example 1: Computer Vision



Deep learning architectures were based on Convolutional Neural Networks (CNN).

Example 2: Machine Translation

Objective : translate a text from a language to another.



Zhang et al., "The AI Index 2021 Annual Report"

- **Oct. 2013**: Pionneering scientic paper (Kalchbrenner, N., and Blunsom, P).
- **2016**: Neural machine translation outperform traditional approaches on public benchmarks
- **2017**: Major systems switch to neural machine translation (using deep recurrent neural networks)

Example 3: Playing Games

- **1997**: Deep Blue defeats Kasparov at Chess.
- **2016**: AlphaGo's victory again Lee Sedol at Go.
- **2017**: AphaGo Zero learns how to play Go only by playing against itself. It outperformed previous AlphaGo version (Reinforcement learning)
- **2017**: DeepStack beats professional human poker players.

Protein folding

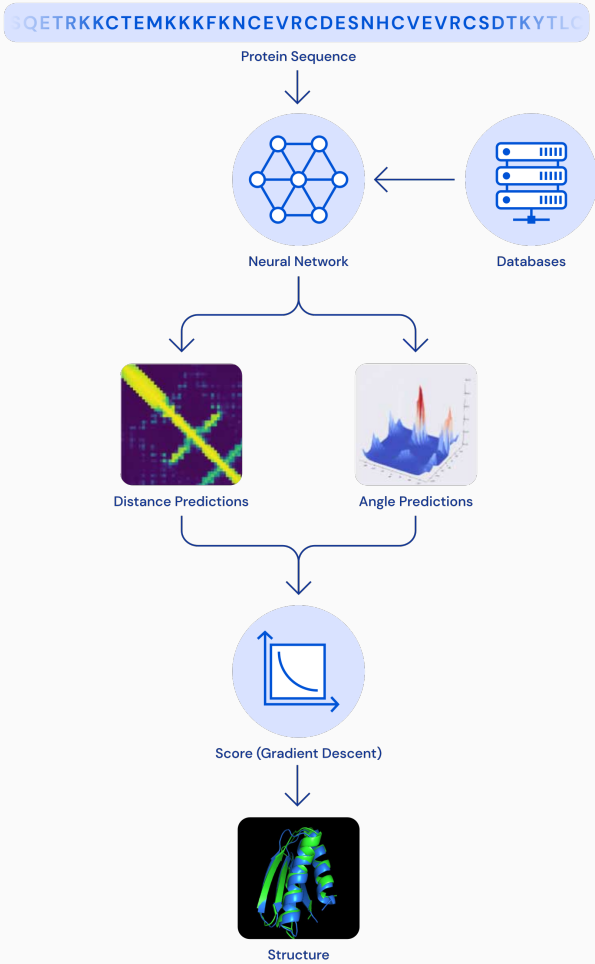


Diagram of Alpha Fold (source: Deepmind)

Senior et al. 2020

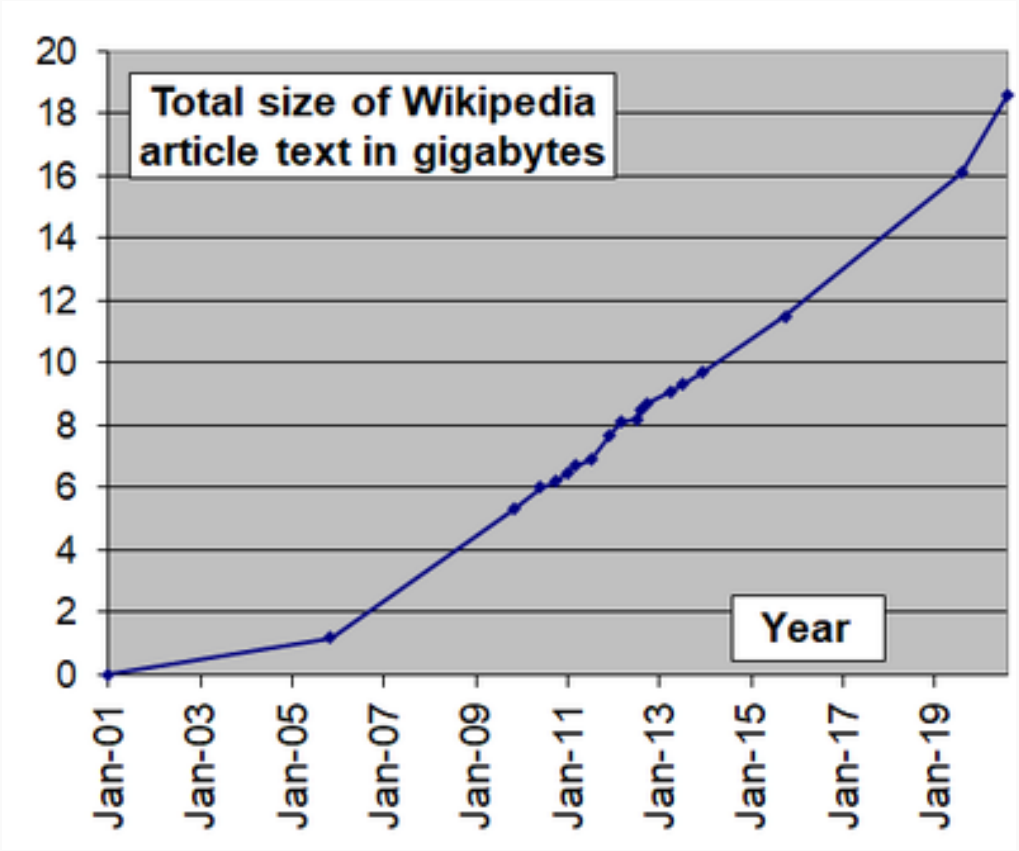
AI Art?

Edmond de Bellamy by Obvious(collective)

Generated using a Generative Adversarial Network.
Selling prince (Oct. 2018): $432,000

Reasons for these recent achievements?

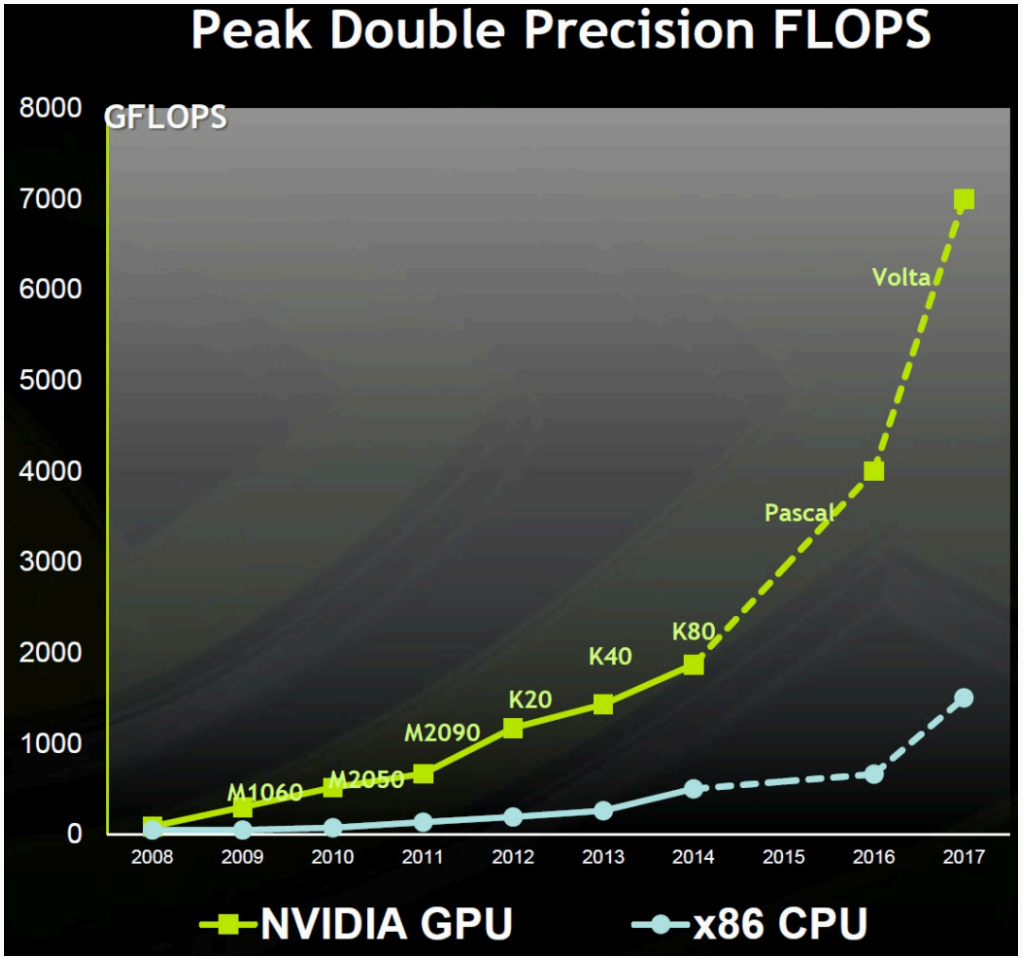
- Increasing of the datasets in size and quality



source: Wikipedia

Reasons for these recent achievements?

- Increasing of the datasets in size and quality
- Progress in computing resources.



source: NVDIA

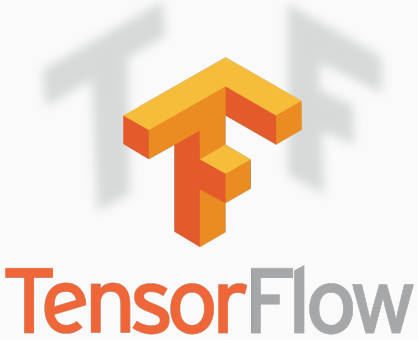
Reasons for these recent achievements?

- Increasing of the datasets in size and quality
- Progress in computing resources.
- Scientific research on new algorithms (e.g adapted to image processing)

source: Deep Dream Generator

Reasons for these recent achievements?

- Increasing of the datasets in size and quality
- Progress in computing resources.
- Scientific research on new algorithms (e.g adapted to image processing)
- Very efficient software (GPU, cloud computing, automatic differentiation, ...)



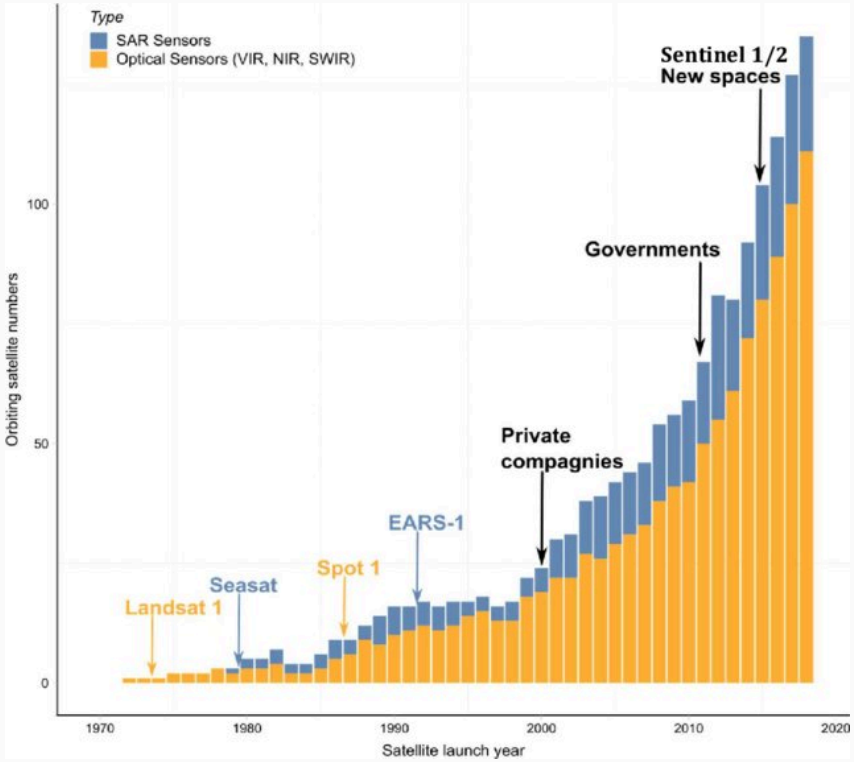
Reasons for these recent achievements?

- Increasing of the datasets in size and quality
- Progress in computing resources.
- Scientific research on new algorithms (e.g adapted to image processing)
- Very efficient software (GPU, cloud computing, automatic differentiation, ...)
- Free software and open data culture.

Apply Machine-Learning to physical (Earth-system) modelling?

Why is it a good idea?

- A increasing number of geophysical data (one spatial mission: 24 TB/day)



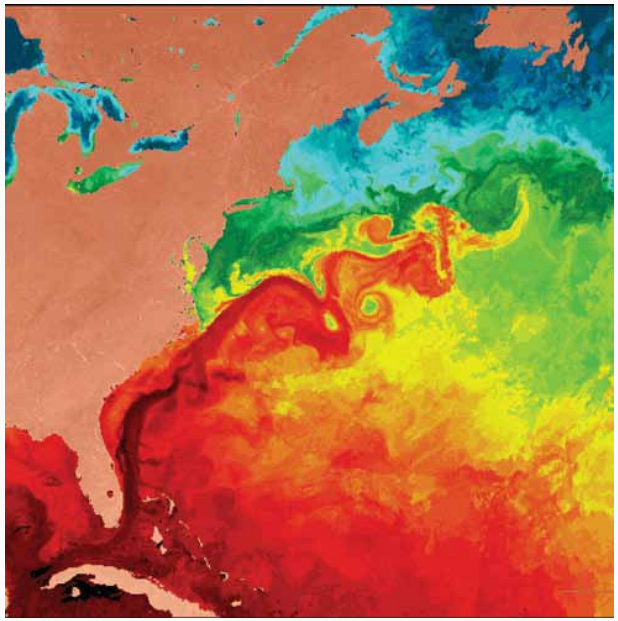
Earth System observavtion satellites

source: Tonneau et al. (2020)

Apply Machine-Learning to physical (Earth-system) modelling?

Why is it a good idea?

- A increasing number of geophysical data (one spatial mission: 24 TB/day)
- Data with highly significant spatial patterns



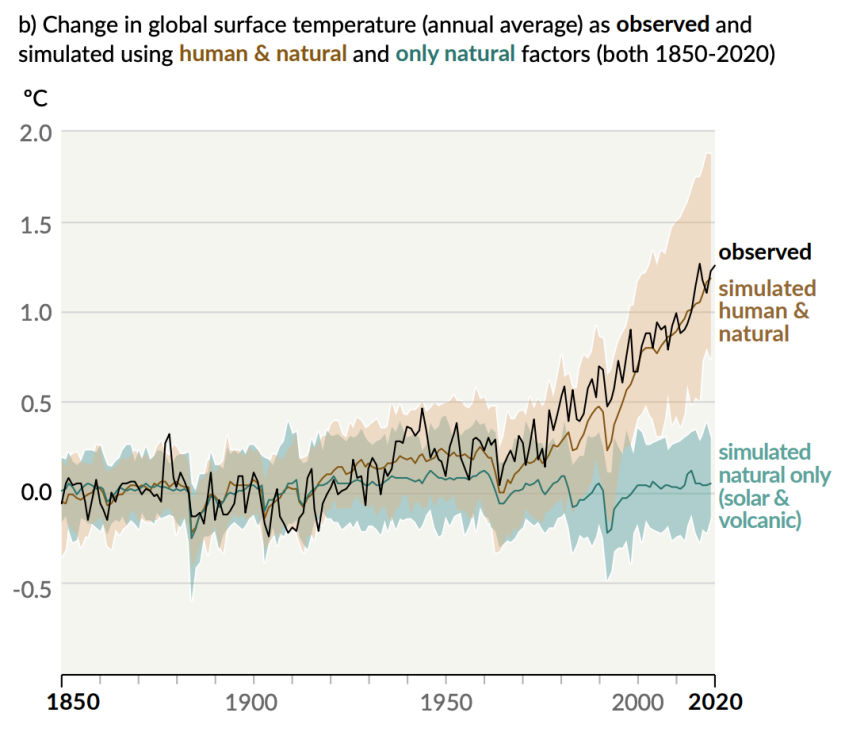
Sea Surface temperature of the gulf stream

source: Talley (2000)

Why is physical modelling specific?

NASDAQ Composite sock market index over the last 10 years

Figure 1: IPCC, AR6, WG1

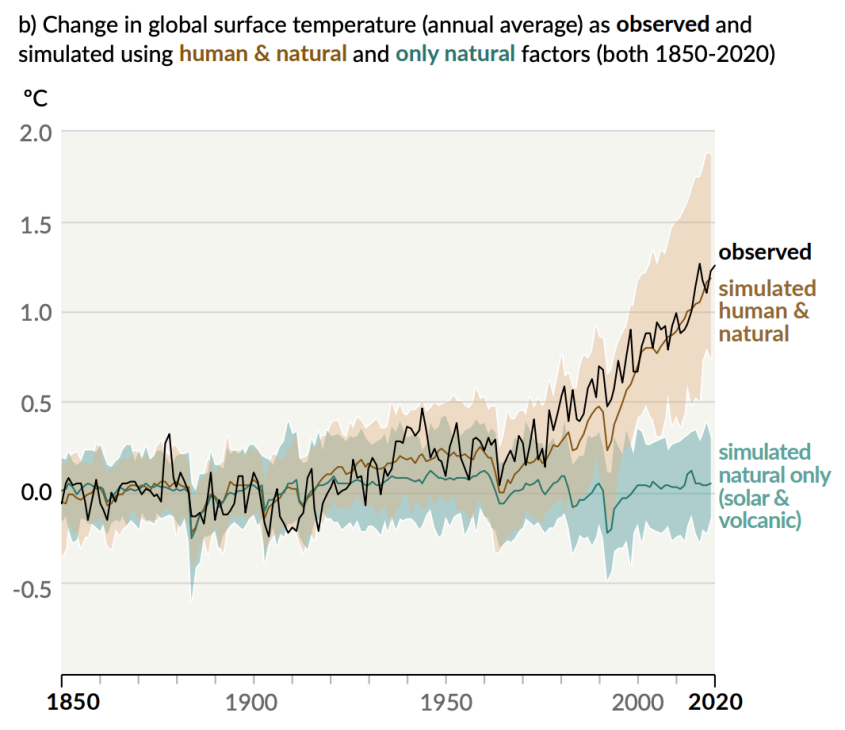


Why is physical modelling specific?

NASDAQ Composite sock market index over the last 10 years

Mostly unknown dynamical processes

Figure 1: IPCC, AR6, WG1



Mostly known dynamical processes (based on physical principles)

What about data assimilation?

Machine learning and data assimilation are closely linked.

Some references:

- Geer, A.J., 2021. Learning earth system models from observations: machine learning or data assimilation?. *Philosophical Transactions of the Royal Society A*, 379(2194)
- Brajard et al. 2019. Connections between data assimilation and machine learning to emulate a numerical model. *Proceedings of the 9th International Workshop on Climate informatics*
- Bocquet et al. 2019. Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlinear processes in geophysics.* 26(3).
- Abarbanel, H.D., Rozdeba, P.J. and Shirman, S., 2018. Machine learning: Deepest learning as statistical data assimilation problems. *Neural computation, 30(8).*

During this summer school:

Marc Bocquet talk on Friday Aug. 27th, 9:30 (CEST).

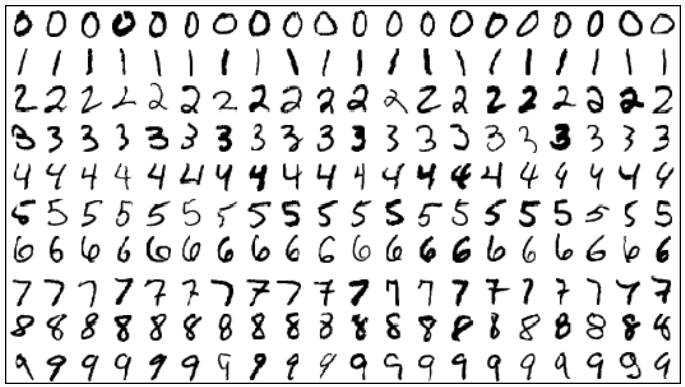
Generalities on Machine Learning

What is this about ?

Can we extract knowledge, make some predictions, determine a ”model” using this large amount of data ?

What is this about ?

Can we extract knowledge, make some predictions, determine a "model" using this large amount of data ?

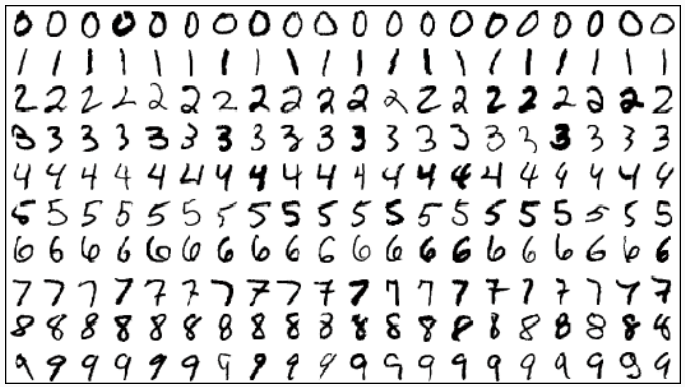


Base of images

$\longrightarrow$ Digite$\in \{0, \ldots, 9\}$

What is this about ?

Can we extract knowledge, make some predictions, determine a "model" using this large amount of data ?



Base of images

$\longrightarrow$ Digit$\in \{0, \ldots, 9\}$

- From high dimensional data (thousands to millions dimensions) to reduced dimensional data (less than 100)
- From disorganized data to comprehensive information
- Can we teach a machine how to do that ?

Two classes of Machine Learning problems

1. Regression: Determination of a quantitative variable from a set of data
   - The price of a building from various predictors (Surface, ...)
   - A physical value (Temperature, humidity, ...) in the future knowing the past
   - ...

Two classes of Machine Learning problems

1. Regression: Determination of a quantitative variable from a set of data
   - The price of a building from various predictors (Surface, ...)
   - A physical value (Temperature, humidity, ...) in the future knowing the past
   - ...
2. Classification: Determination of a class
   - A digit from a image
   - Identification of the content of an image
   - ...

Two types of objectives

1. Supervised learning: we have a set of labeled data with examples of targets.

Two types of objectives

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a ”useful” representation of these data, or some coherent categories.

Two types of objectives

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a "useful" representation of these data, or some coherent categories.
    - Determine typical behaviors of clients in a supermarket knowing what the have bought.

Two types of objectives

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a "useful" representation of these data, or some coherent categories.
    - Determine typical behaviors of clients in a supermarket knowing what the have bought.
3. Semi-Supervised Learning: Only a few subset of the data are labeled

Two types of objectives

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a "useful" representation of these data, or some coherent categories.
    - Determine typical behaviors of clients in a supermarket knowing what the have bought.
3. Semi-Supervised Learning: Only a few subset of the data are labeled
4. Reinforcement Learning: We can initiate and observe the interaction of an agent with its environment. We want to optimize the behavior of the agent.

Two types of objectives

1. Supervised learning: we have a set of labeled data with examples of targets.
2. Unsupervised learning: we only have unlabeled the data, we have no examples of what we want to obtain. We want to extract a "useful" representation of these data, or some coherent categories.
    - Determine typical behaviors of clients in a supermarket knowing what the have bought.
3. Semi-Supervised Learning: Only a few subset of the data are labeled
4. Reinforcement Learning: We can initiate and observe the interaction of an agent with its environment. We want to optimize the behavior of the agent.
    - Playing a chess game.

Formally

A Machine

$$y = \mathcal{M}(x, \theta)$$

- x: input
- y: output
- $\mathcal{M}$: a model (named "machine")
- θ : parameters of the model $\mathcal{M}$.

Machine learning consists in optimizing θ using a set of data. This is the training process.

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - x and some subset of y: semi-supervised learning

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - x and some subset of y: semi-supervised learning
- An objective
  - y is quantitative: regression
  - y is a class: classification

The Machine Learning recipe

A Machine

$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - x and some subset of y: semi-supervised learning
- An objective
  - y is quantitative: regression
  - y is a class: classification
- A computational architecture (the machine)
  - linear
  - non-linear
  - neural networks, random forest, ...

The Machine Learning recipe

A Machine

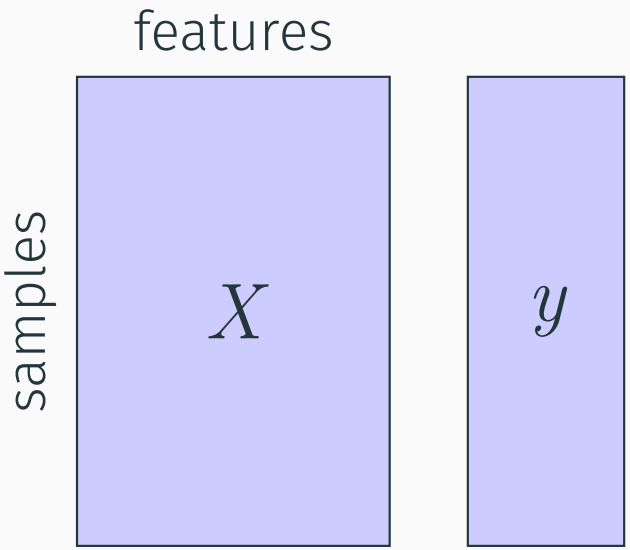
$$y = \mathcal{M}(x, \theta)$$

What are the ingredients?

- Some data
  - x, y : supervised learning
  - only x: unsupervised learning
  - x and some subset of y: semi-supervised learning
- An objective
  - y is quantitative: regression
  - y is a class: classification
- A computational architecture (the machine)
  - linear
  - non-linear
  - neural networks, random forest, ...
- A learning process
  - Estimation of θ

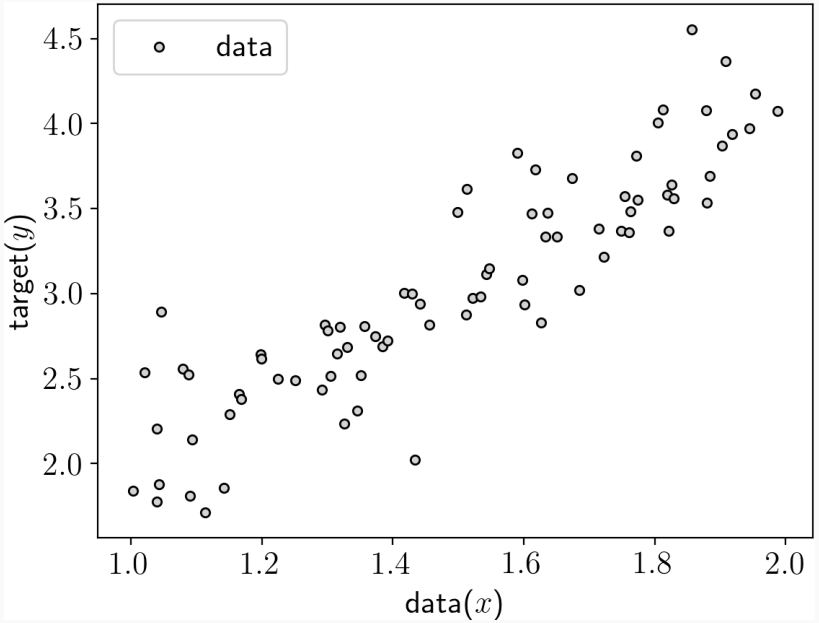
Multidimensional data

Generally, we have multidimensional data X and a one-dimensional target y.



An illustration

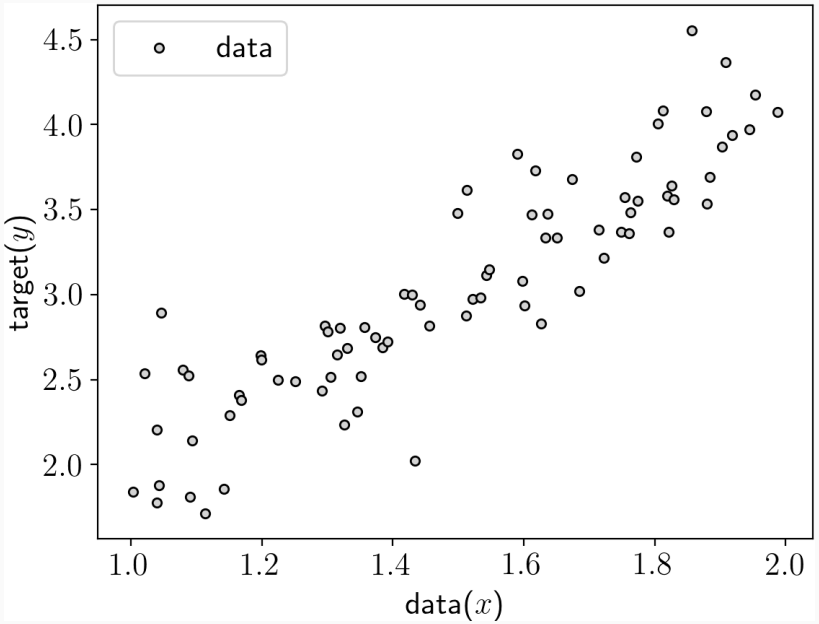
- Some Data



  - there are labeled y: supervised learning
  - y is quantitative: regression

An illustration

- Some Data

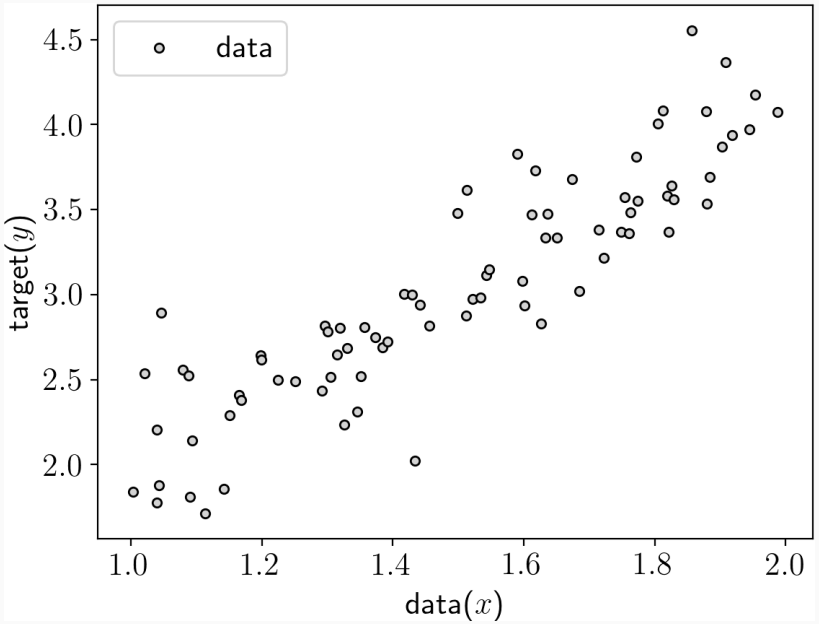


  - there are labeled y: supervised learning
  - y is quantitative: regression

- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

An illustration

- Some Data



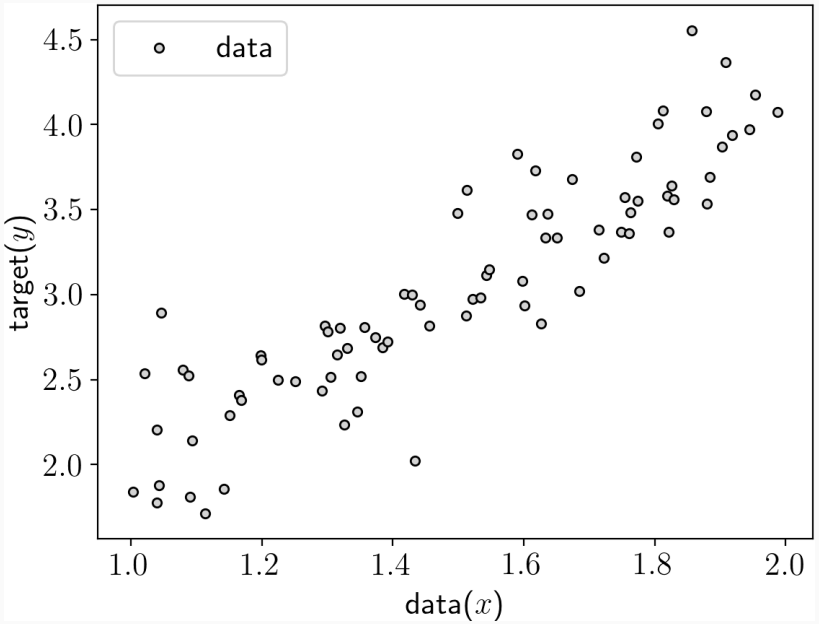
  - there are labeled y: supervised learning
  - y is quantitative: regression

- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

- A model: $\hat{y} = \theta_1 X + \theta_0$ (linear)

An illustration

- Some Data

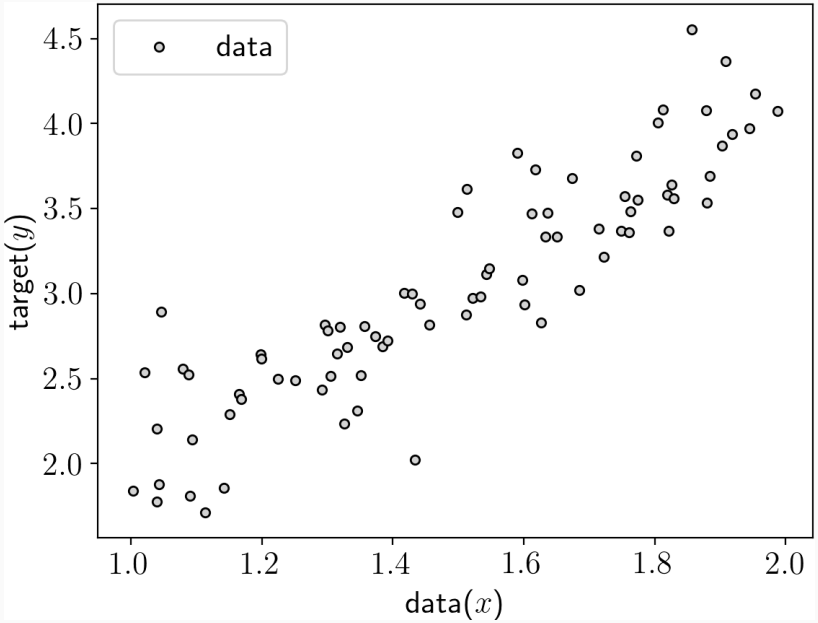


  - there are labeled y: supervised learning
  - y is quantitative: regression
- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

- A model: $\hat{y} = \theta_1 X + \theta_0$ (linear)
- A learning process: $\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$

An illustration

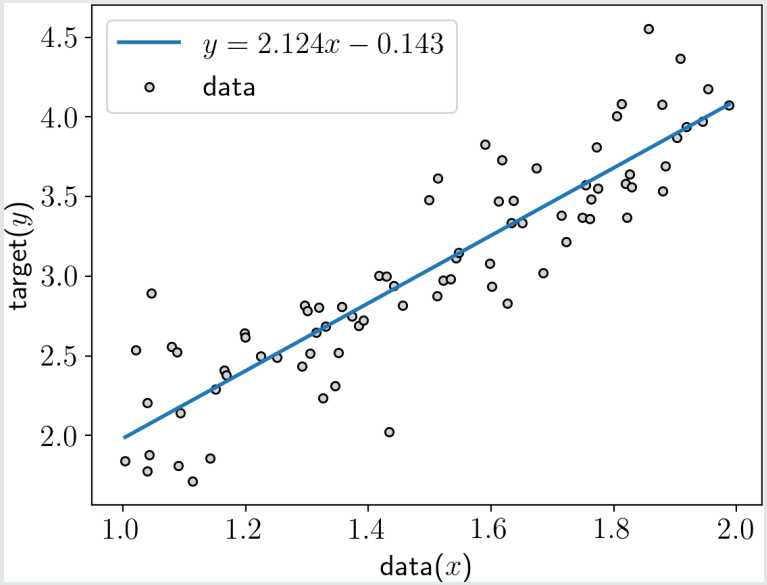
- Some Data



  - there are labeled y: supervised learning
  - y is quantitative: regression
- An Objective: Estimate $\hat{y}$ from x by minimizing $(\hat{y} - y)^2$ (Least-square objective)

- A model: $\hat{y} = \theta_1 X + \theta_0$ (linear)
- A learning process: $\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$

Result



Model selection/validation

Choice of the model

Polynomial regression

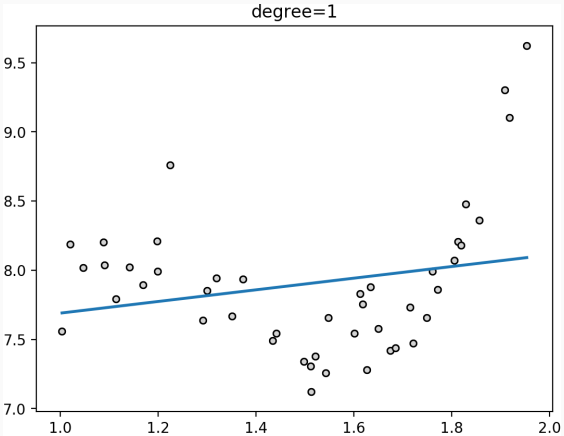
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

Choice of the model

Polynomial regression

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

degree = 1 (linear)

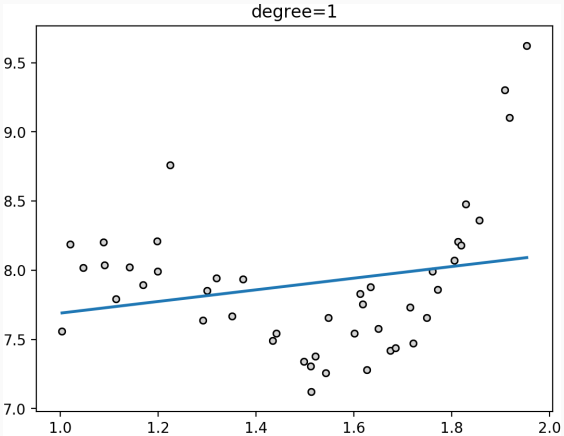


Choice of the model

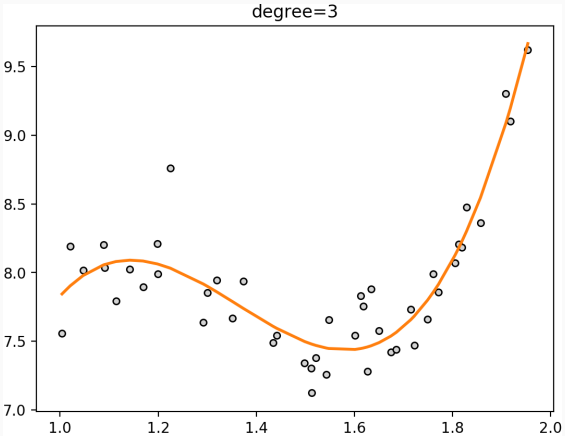
Polynomial regression

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

degree = 1 (linear)



degree = 3

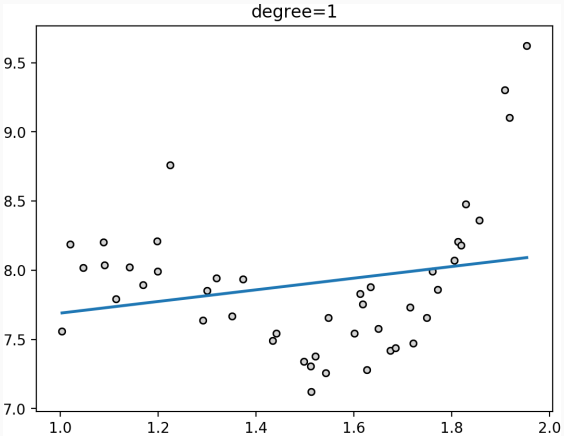


Choice of the model

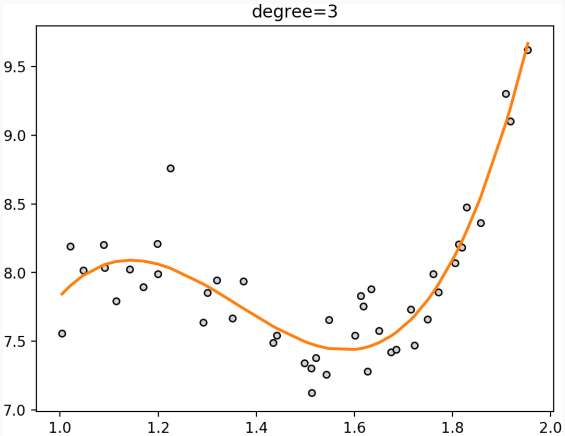
Polynomial regression

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

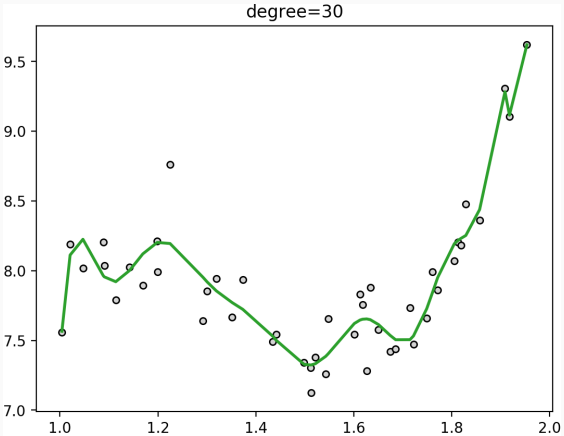
degree = 1 (linear)



degree = 3



degree = 30



What is the best model?

Train/Validation split

The idea

Evaluate a score on a independent dataset

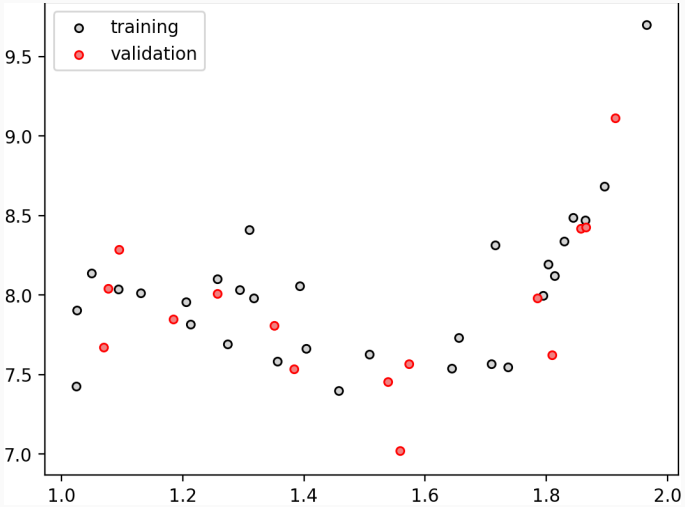
Train/Validation split

The idea

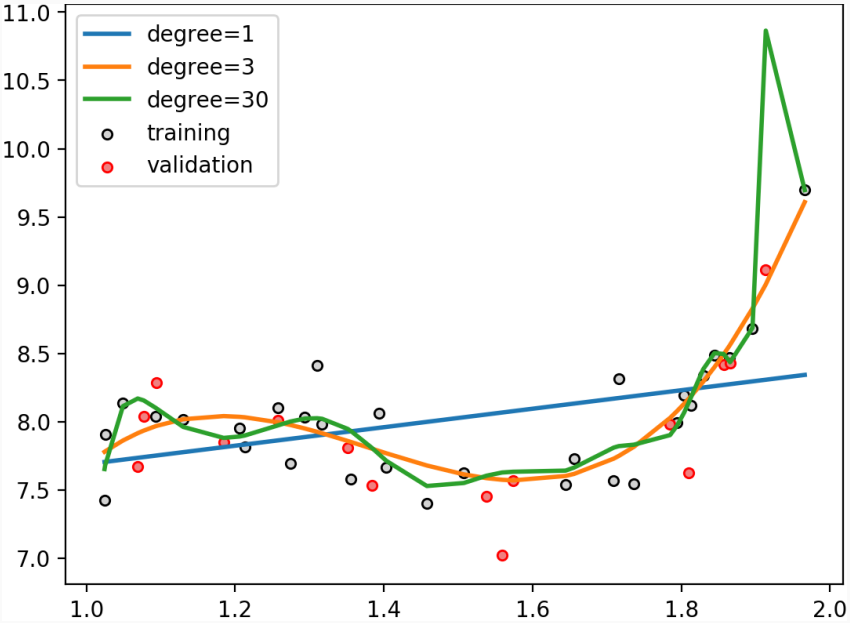
Evaluate a score on a independent dataset

In our example we can randomly divide (X, y) in two datasets:

- The training dataset X_{train}, y_{train} used to fit the model.
- The validation dataset X_{val}, y_{val} used to compute the score (e.g., correlation, mean-squared error)



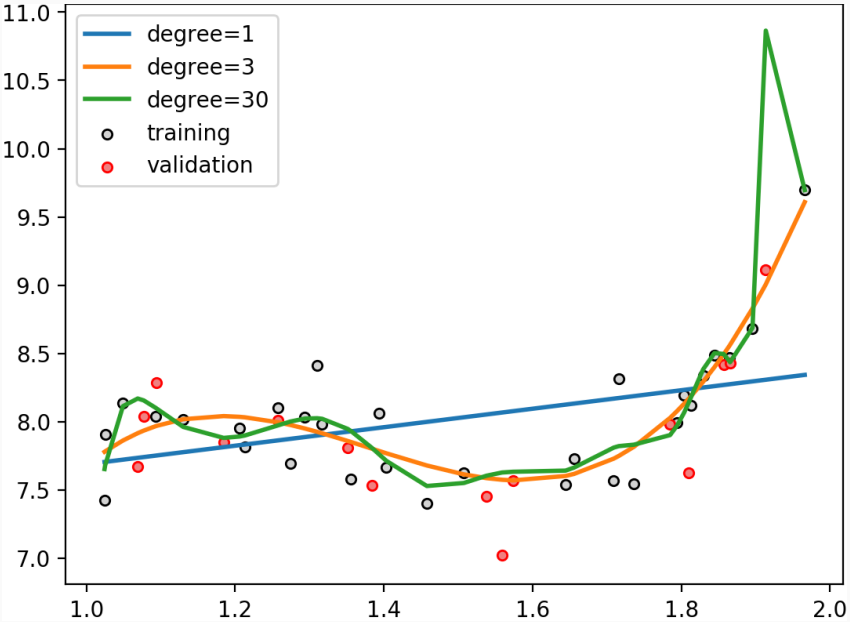
Choice of the model



Score: Mean Square Error (MSE)

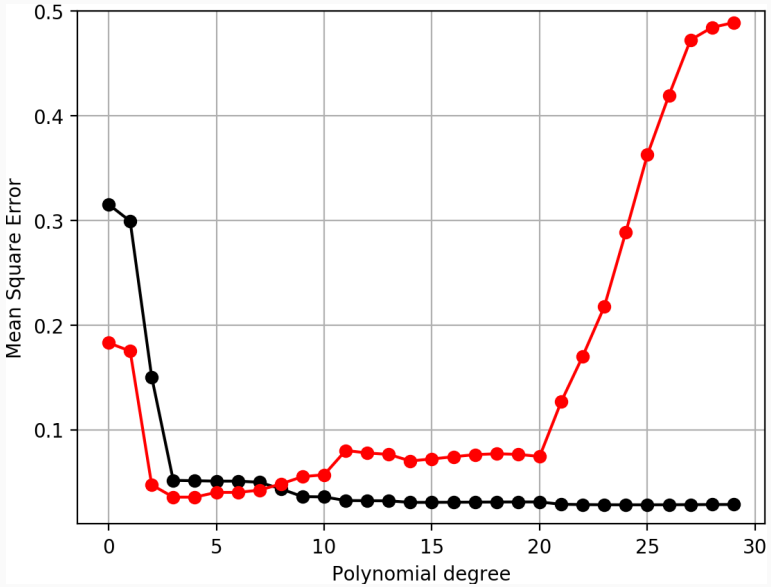
Deg.	Train Score	Val. Score
1	0.17	0.23
3	0.045	0.062
30	0.035	0.27

Choice of the model



Score: Mean Square Error (MSE)

Deg.	Train Score	Val. Score
1	0.17	0.23
3	0.045	0.062
30	0.035	0.27

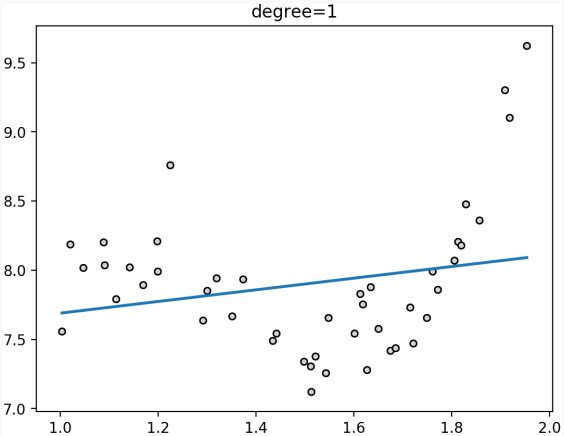


Choice of the model

Polynomial regression

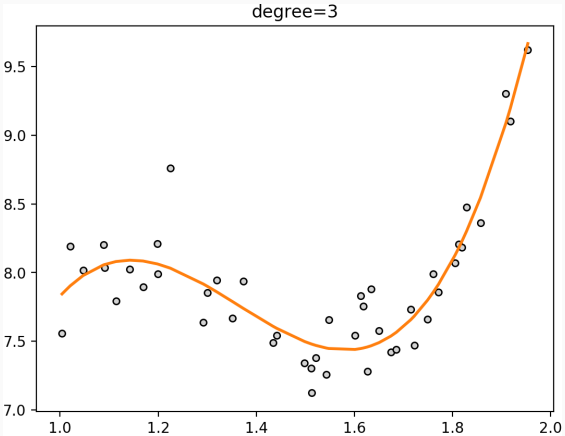
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d = \sum_{i=0}^{d} \theta_i X^i$$

degree = 1 (linear)



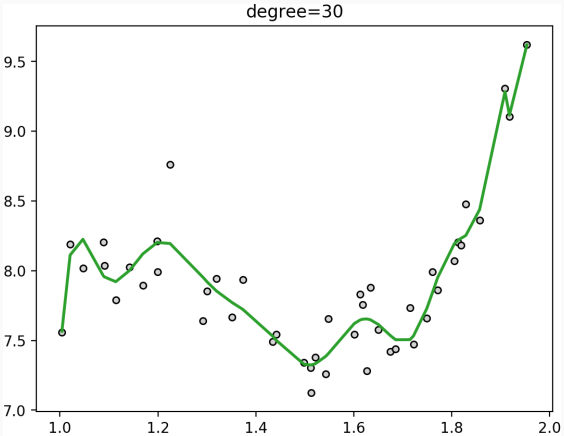
underfitting

degree = 3



good fit

degree = 30



overfitting

Train/Validation split

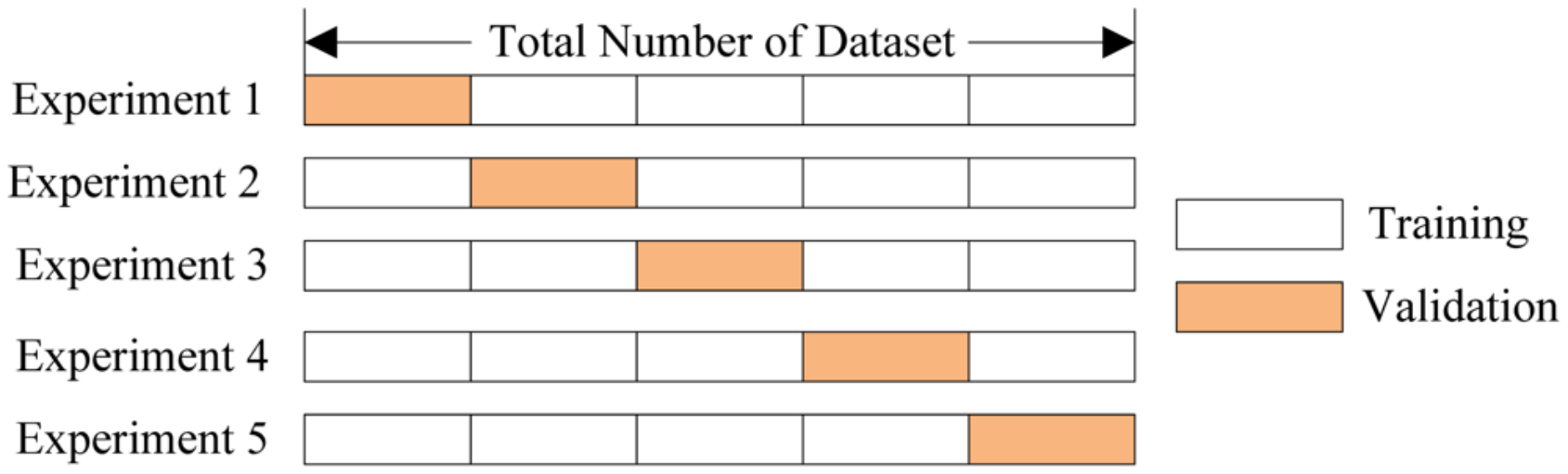
Drawbacks

- drastically reduce the number of samples which can be used for learning the model
- Results can depend on a particular random choice for the pair of (train, validation) sets.

More Robust: cross validation

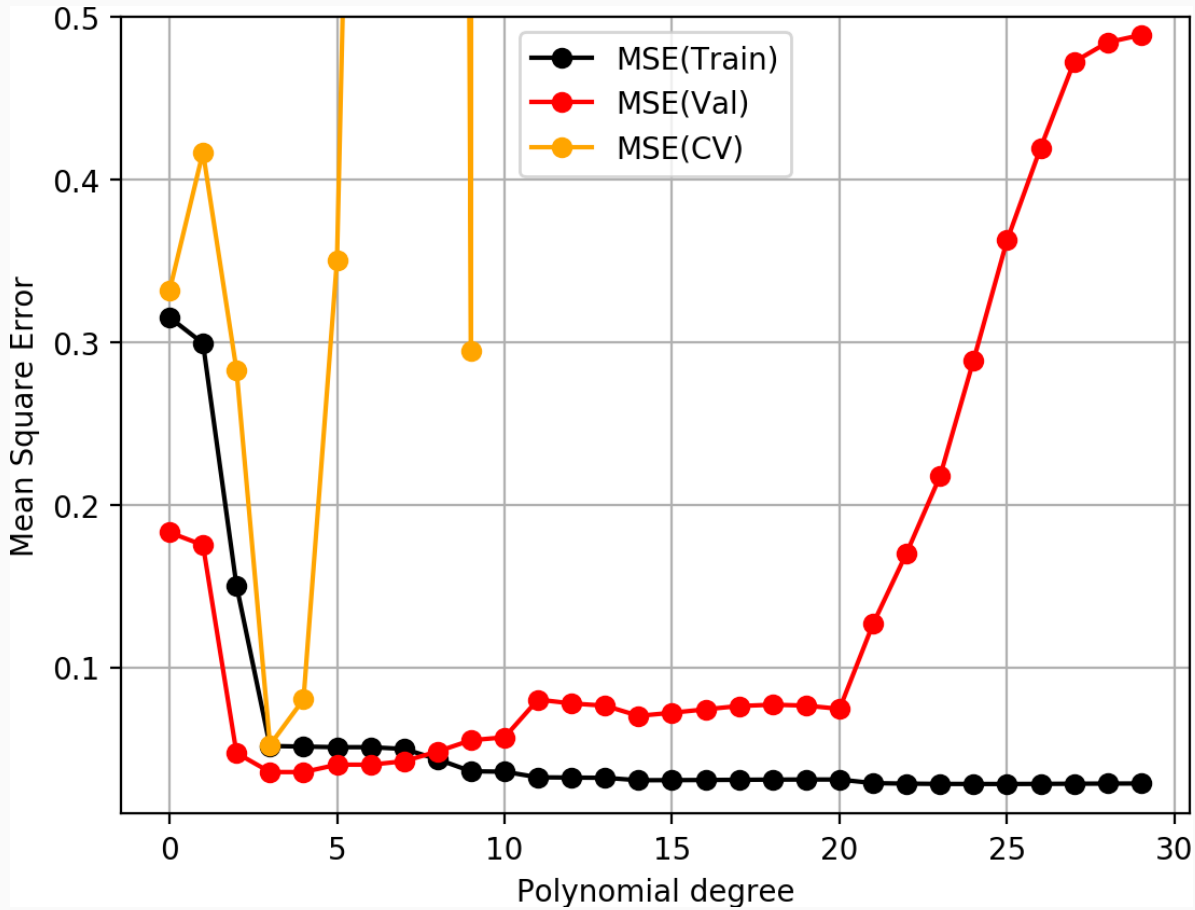
The idea

- Dividing the data in n folds,
- Learning n model (each time with a different training set),
- Compute the mean score over n validation set.



Cross-Validation

Fold	MSE
1	0.052
2	0.043
3	0.137
4	0.025
5	0.048
6	0.144
7	0.011
8	0.025
9	0.010
10	0.028
Mean	0.05



Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).

Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.

Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.
3. But then... the validation set was used to determine the best machine learning process

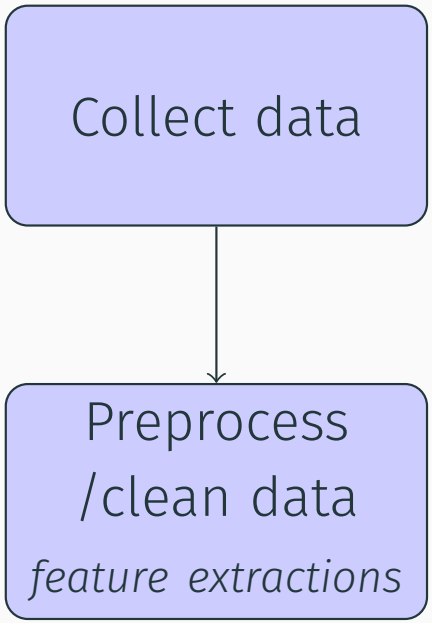
Wrapping up

1. When applying machine learning techniques there are hyperparameters to be determined (e.g., degree of the polynomial in polynomial regression).
2. These hyperparameters can be determined by splitting the data into training/validation or by cross-validation.
3. But then... the validation set was used to determine the best machine learning process
4. To evaluate independanly the performance of our model, we should compute the score on a third independant dataset: The test dataset.

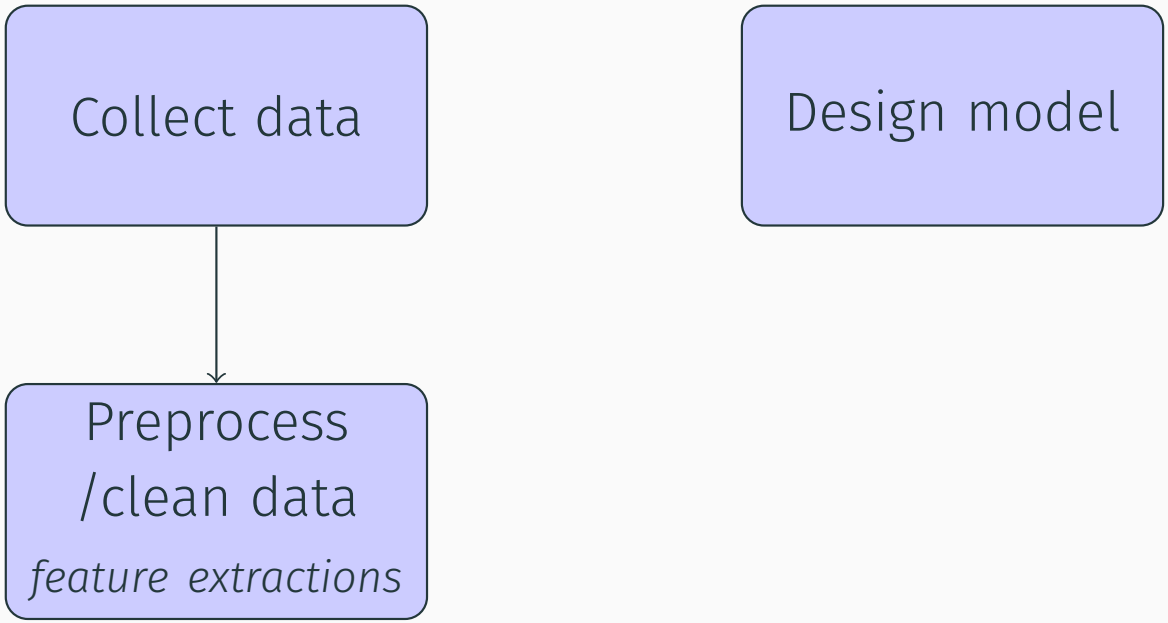
Steps of a machine learning process

Steps

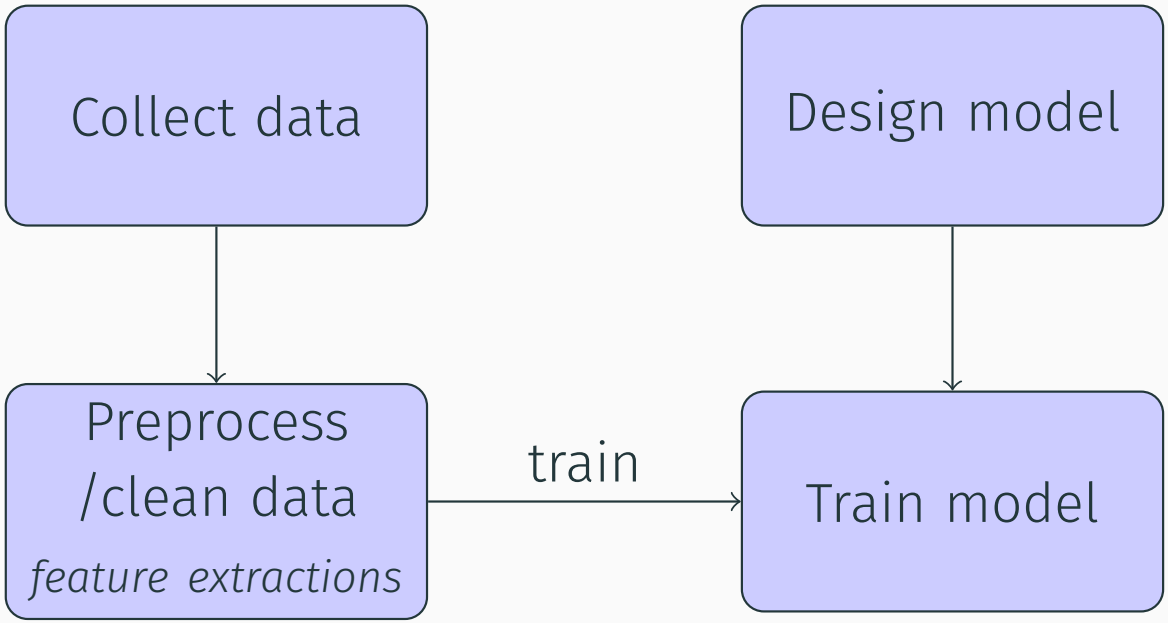
Steps



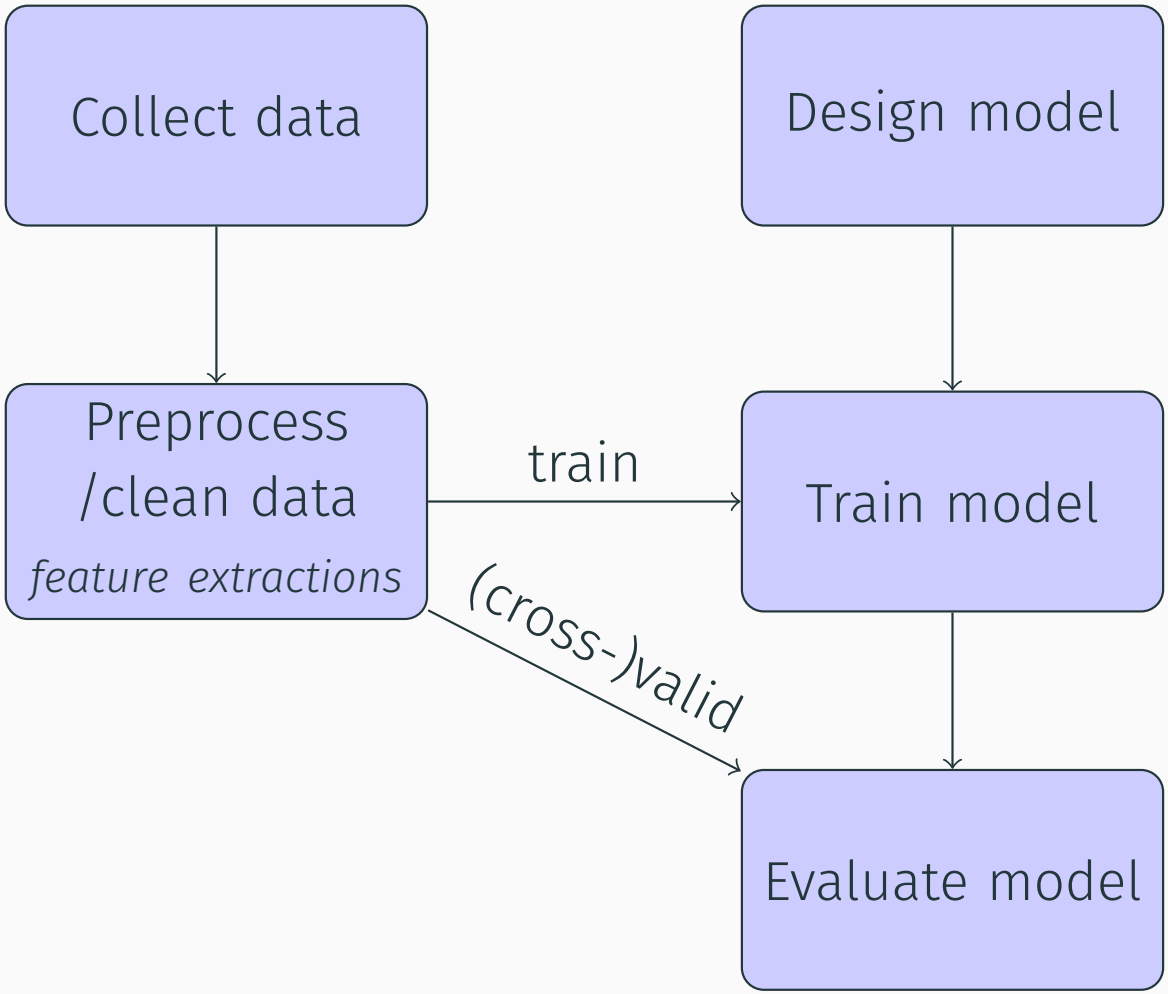
Steps


Collect data
Preprocess
/clean data
feature extractions
Design model

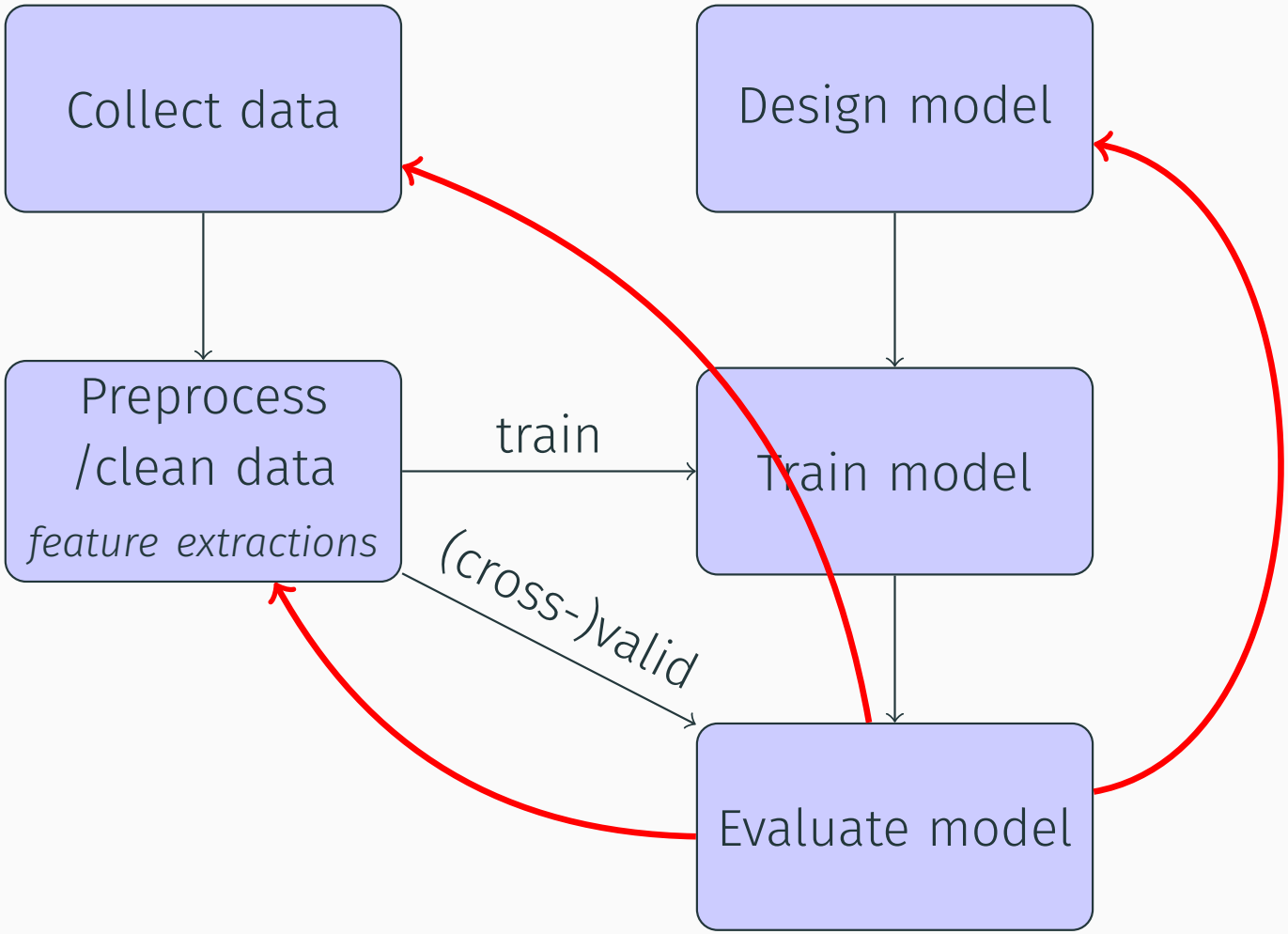
Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model

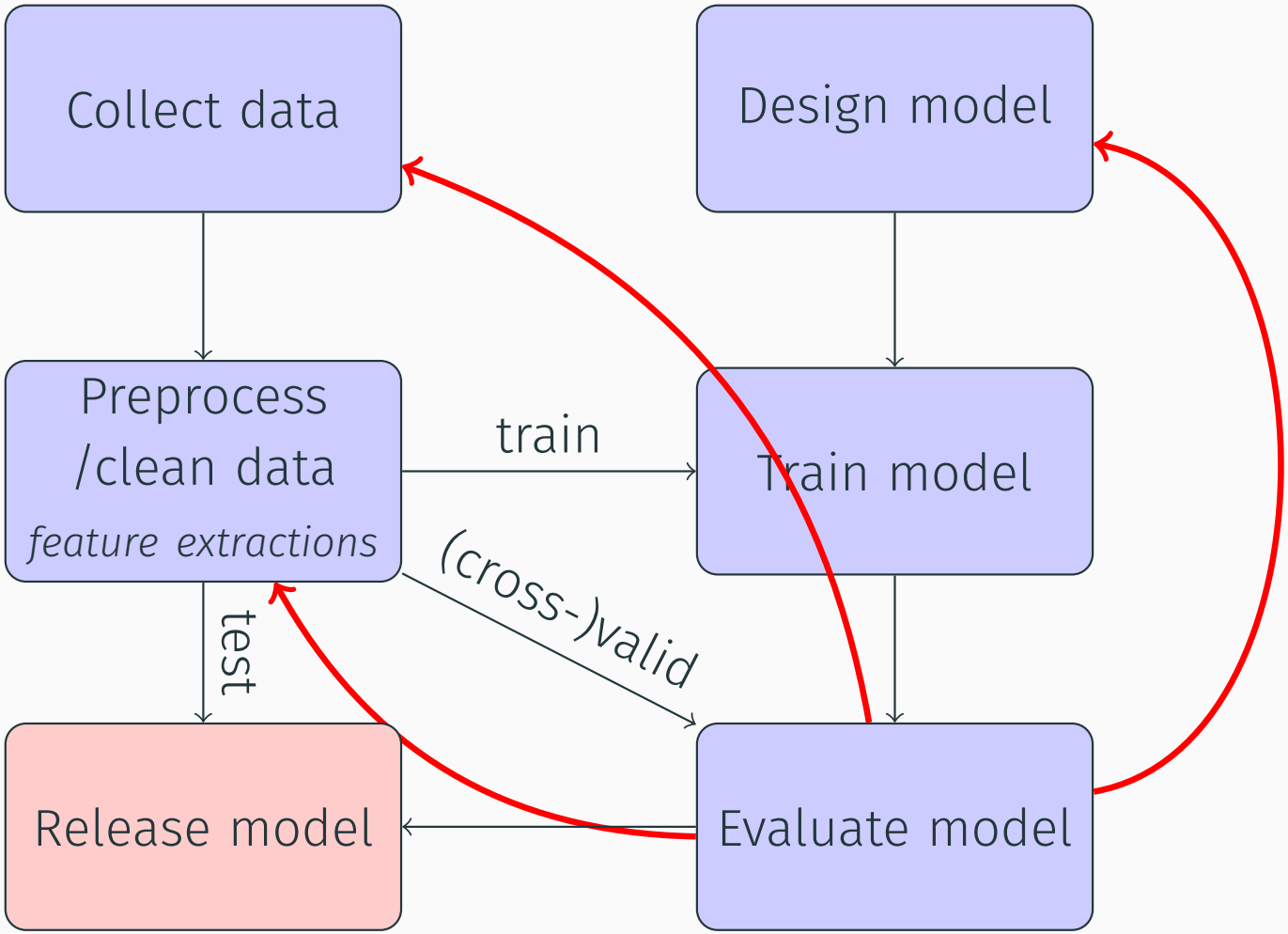
Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model
(cross-)valid
Evaluate model

Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model
(cross-)valid
Evaluate model

Steps


Collect data
Design model
Preprocess
/clean data
feature extractions
train
Train model
(cross-)valid
test
Release model
Evaluate model

In summary

From one dataset, 3 sub-datasets have to be extracted:

- A training dataset
- A validation dataset

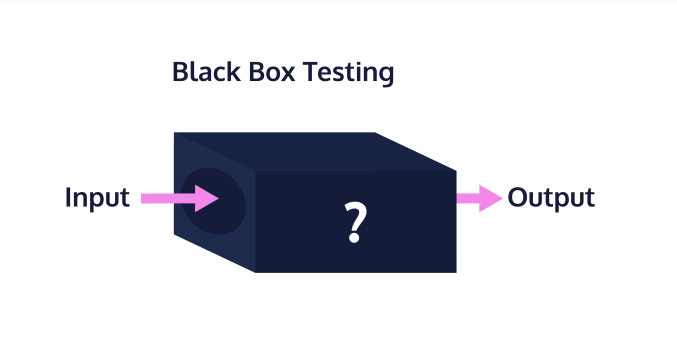
Can be done iteratively in a cross-validation procedure.
Some parameters of the model (e.g. polynomial order in a polynomial regression) were determined from the validation dataset.

- A test dataset (independent from the two other) to estimate the final performance of the model.

A black box?

The black box paradigm

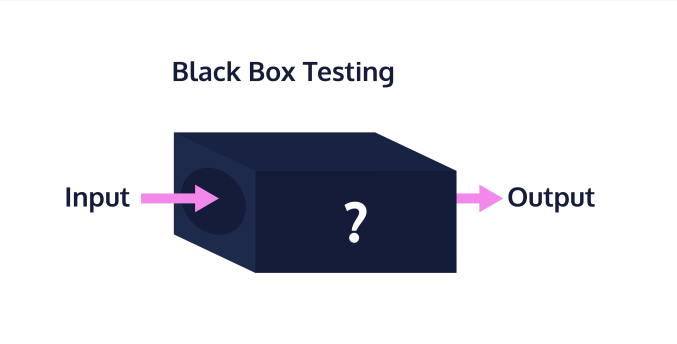
The machine-learning based model is a black box. It gives some results but we don't understand how.



Do we need to understand the model?

The black box paradigm

The machine-learning based model is a black box. It gives some results but we don't understand how.



Do we need to understand the model?

"Every time I fire a linguist, the performance of our speech recognition system goes up."

F. Jelinek, 1988

Frederick Jelinek 1932-2010

Beyond the black box

Motivation

- Build models that can be trusted
- Use other source of knowledge (e.g. physical properties) when data are not sufficient.

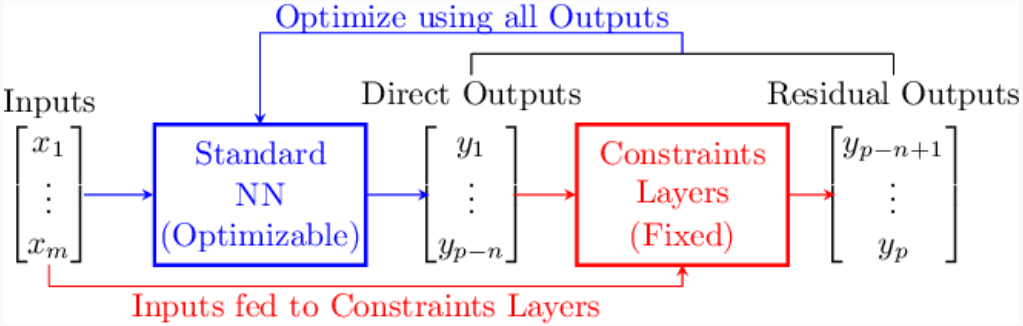
Two directions:

- Add physical constraints to ML models
- Explainable/Transparent ML.

Add physical constraints to ML models

- Simple example: enforce the positivity of some quantities (e.g. concentration)
- More complex: enforce conservation laws

  Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P. and Gentine, P., 2021. Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, 126(9), p.098302.

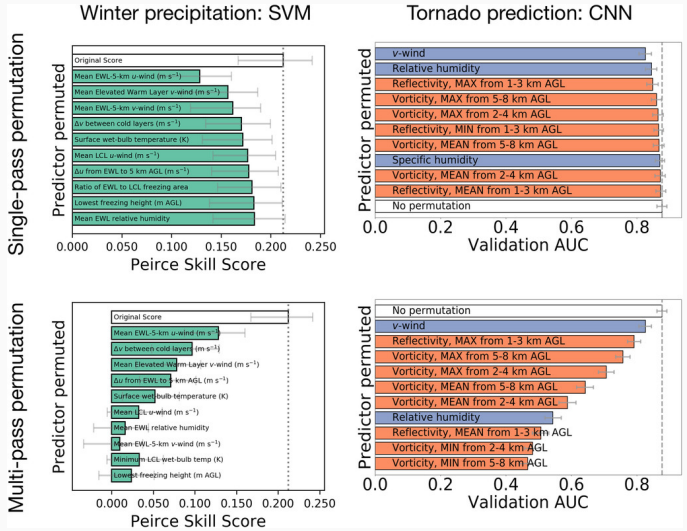


Beucler et al.

Explainable/Transparent ML.

The objective is to understand how the machine learning makes a prediction (e.g. which feature is important for the prediction).

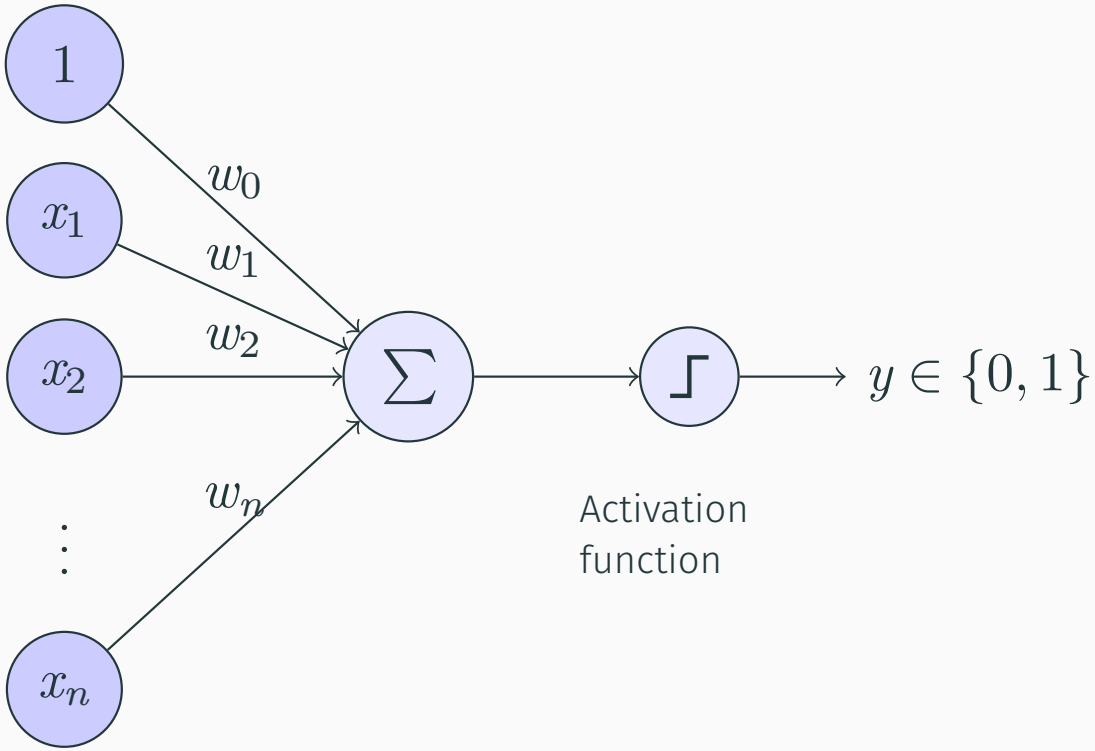
- McGovern, A., Lagerquist, R., Gagne, D.J., Jergensen, G.E., Elmore, K.L., Homeyer, C.R. and Smith, T., 2019. Making the black box more transparent: Understanding the physical implications of machine learning. *Bulletin of the American Meteorological Society, 100(11)*, pp.2175-2199.
- Sonnewald, M., Lguensat, R., Jones, D. C., Dueben, P. D., Brajard, J., and Balaji, V.,2021. Bridging observation, theory and numerical simulation of the ocean using Machine Learning. *Environ. Res. Lett.* 16 073008



McGovern et al.

Neural Networks

The perceptron : an artificial neuron



Rosenblatt, 1957

Computation

$$y = f(w_0 + w_1.x_1 + w_2.x_2 + \cdots + w_n.x_n) = f\left(w_0 + \sum_{i=1}^{n} w_i.x_i\right)$$

Some remarks

- Inputs x_i are the different features of the data

Some remarks

- Inputs x_i are the different features of the data
- Weight w_i are the parameters of the model to optimize

Some remarks

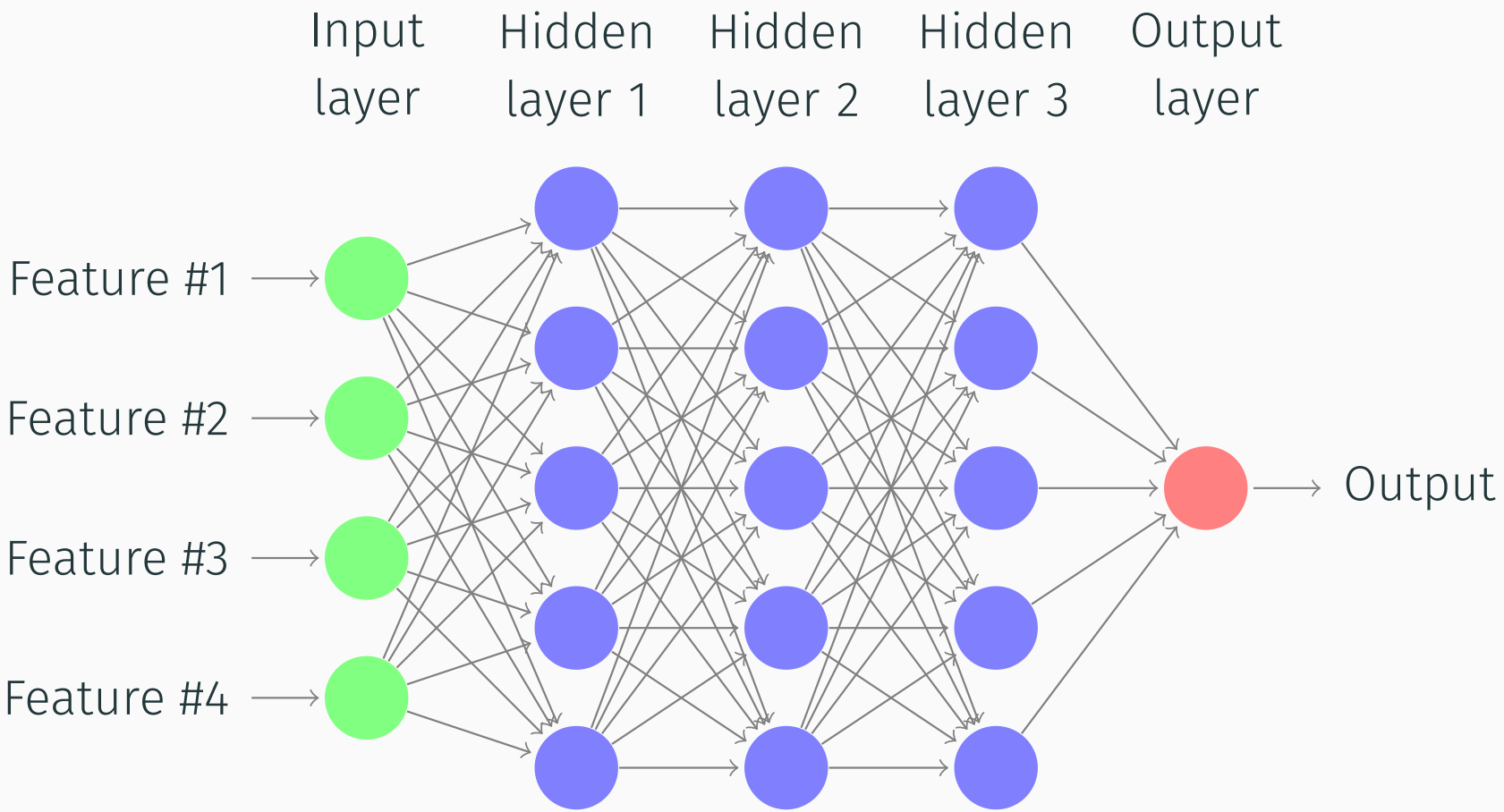
- Inputs x_i are the different features of the data
- Weight w_i are the parameters of the model to optimize
- If the activation function is identity, it is equivalent to a linear regression

Some remarks

- Inputs x_i are the different features of the data
- Weight w_i are the parameters of the model to optimize
- If the activation function is identity, it is equivalent to a linear regression

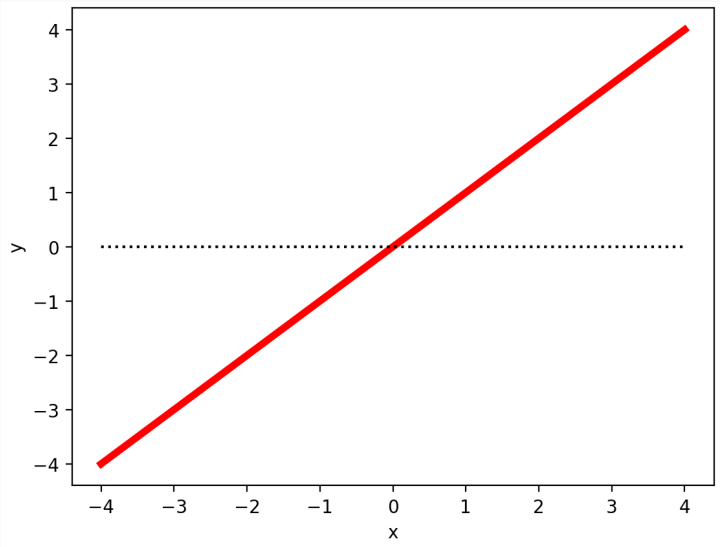
More complexe models are build by combining several perceptrons

Multi-layer perceptron (Densely connected layers)

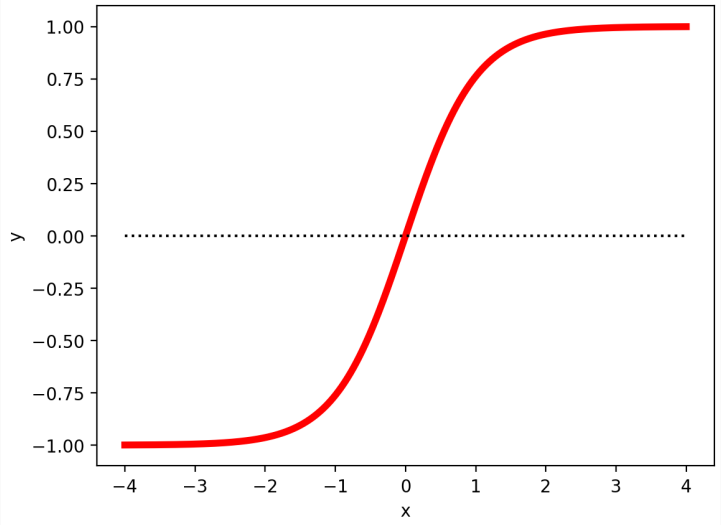


More usual activation functions

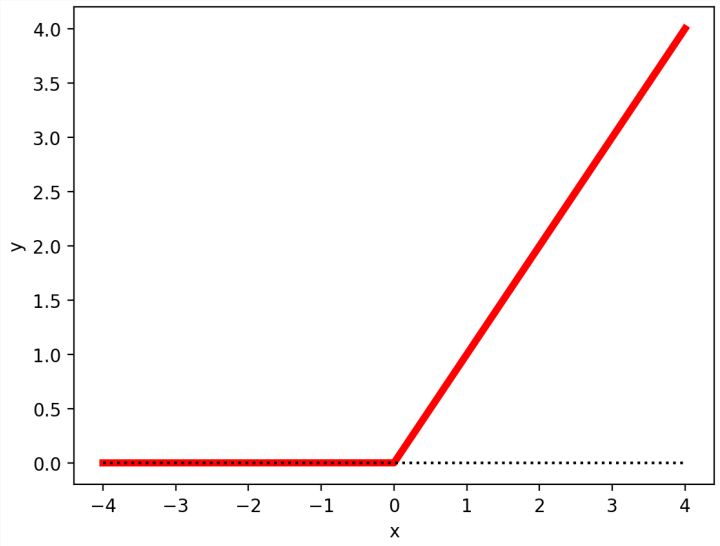
Linear



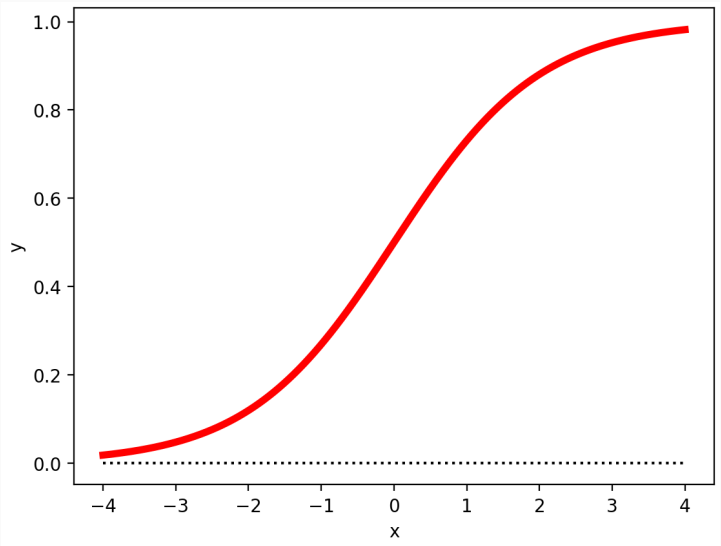
Hyperbolic tangent



ReLU

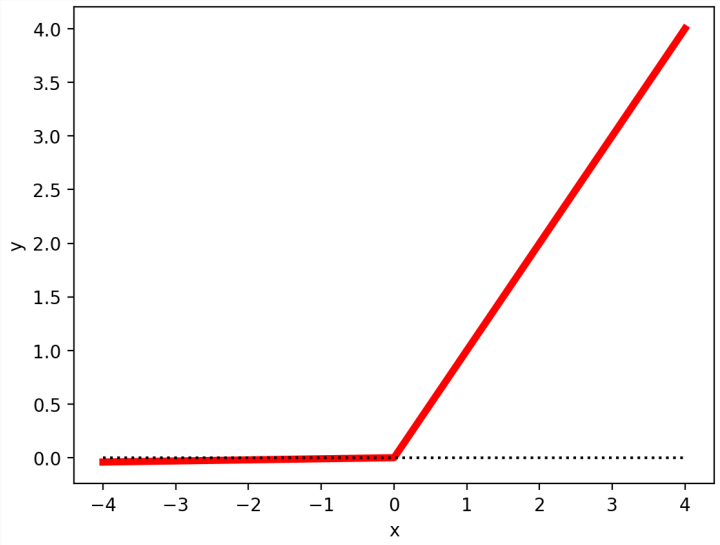


Sigmoid

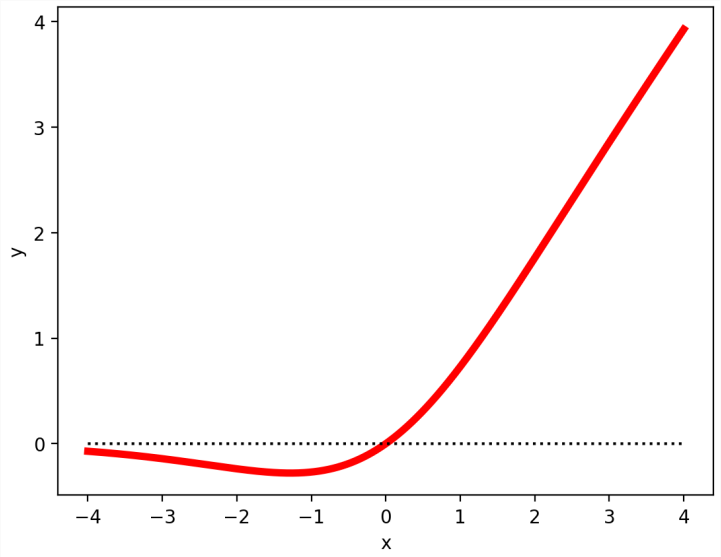


Other ”fancy” activation functions

Leaky-ReLU



Swish



Classification and regression loss

Regression

- Last layer:
  linear or hyperbolic tangent
- Loss function:

$$L(\hat{y}, y) = \sum_i (\hat{y}_i - y_i)^2$$

Classification and regression loss

Regression

- Last layer:
  linear or hyperbolic tangent
- Loss function:

$$L(\hat{y}, y) = \sum_i (\hat{y}_i - y_i)^2$$

Classification

- Last layer:
  Soft-max

$$p_j = f_j(\mathbf{h}) = \frac{e^{h_j}}{\sum_k e^{h_k}}$$

- Loss function:
  Negative crossentropy

$$L(p, y) = -\sum_i \sum_j y_{i,j}. \log p_{i,j}$$

Probabilistic interpretation

Maximum likelihood estimator and loss

We can assume that the observation y follows a gaussian law:

$$p(y/x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(y-\mu(x))^2}{2\sigma^2},$$

where x is observed and $\mu(x)$ is a function of x.
Given a set of samples $(x_k, y_k)_{1:K}$, the negative log-likelihood is defined by

$$L = \sum_{k=1}^{K} \left(\frac{\log 2\pi\sigma^2}{2} + \frac{(y_k - \mu(x_k))^2}{2\sigma^2} \right)$$

Minimizing L is maximizing the probability of having the observations y_k given x_k.

Loss function of a neural net

First case: σ is constant

$\mu(x)$ is parametrized by a neural net $G_\mu(x, \boldsymbol{\theta}_\mu)$
The Maximum likelihood estimator is found by minizming:

$$L(\boldsymbol{\theta}_\mu) = \sum_{k=1}^{K} (y_k - G_\mu(x, \boldsymbol{\theta}_\mu))^2$$

which is exactly the regression loss already introduced.

Loss function of a neural net

First case: σ is constant

$\mu(x)$ is parametrized by a neural net $G_\mu(x, \boldsymbol{\theta}_\mu)$
The Maximum likelihood estimator is found by minizming:

$$L(\boldsymbol{\theta}_\mu) = \sum_{k=1}^{K} (y_k - G_\mu(x, \boldsymbol{\theta}_\mu))^2$$

which is exactly the regression loss already introduced.

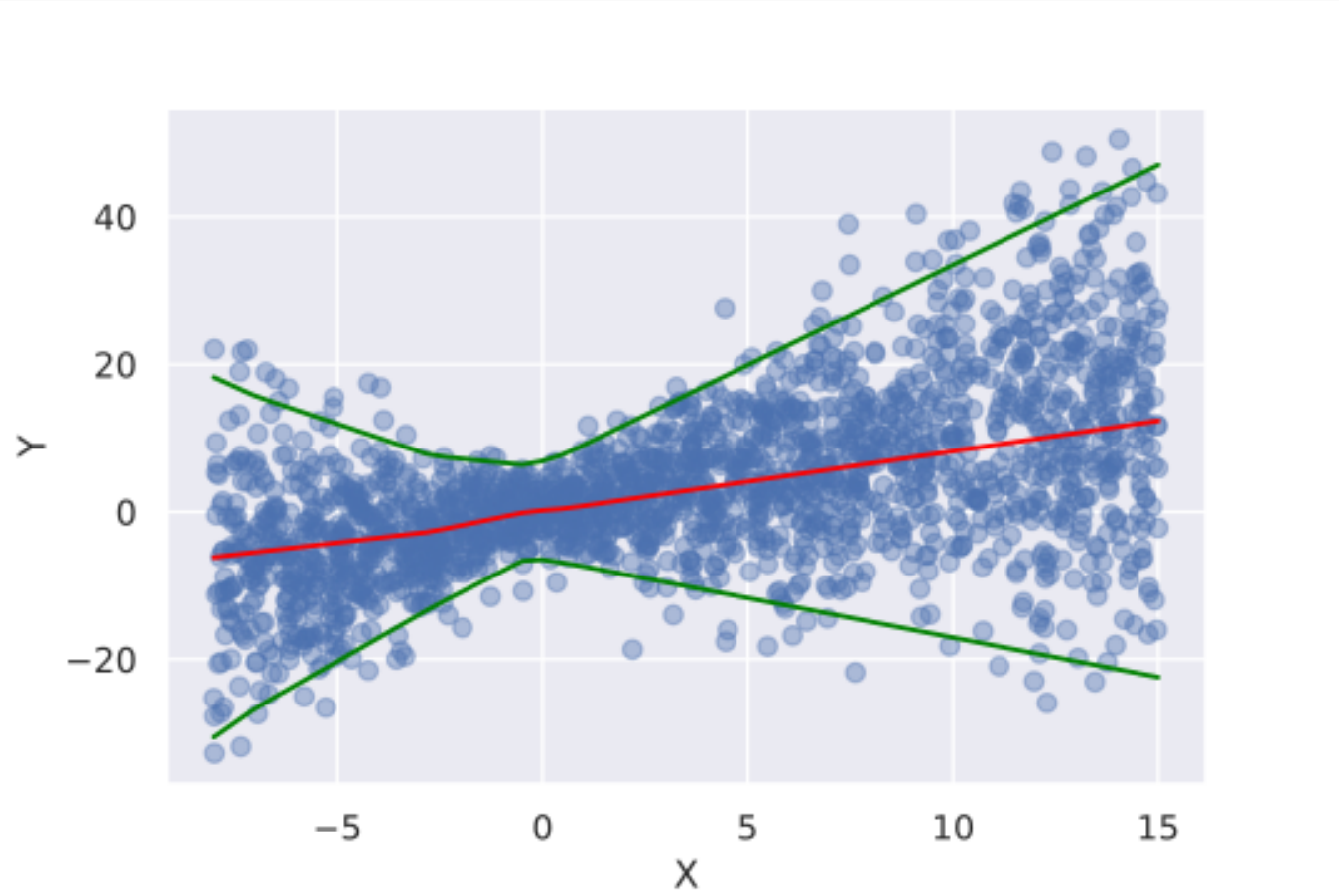
Second case: $\sigma(x)$ is a function of x.

In addition to $G_\mu(x, \boldsymbol{\theta}_\mu)$, $\sigma(x)$ is parametrized by $G_\sigma(x, \boldsymbol{\theta}_\sigma)$. The loss to minimize is then:

$$L(\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma) = \sum_{k=1}^{K} \left(\frac{\log 2\pi\, G_\sigma(x_k, \boldsymbol{\theta}_\sigma)^2}{2} + \frac{(y_k - G_\mu(x_k, \boldsymbol{\theta}_\mu))^2}{2 G_\sigma(x_k, \boldsymbol{\theta}_\sigma)^2} \right)$$

The neural net $G = (G_\mu, G_\sigma)$ gives also the uncertainty of its estimation in the form of the standard deviation.

Illustration

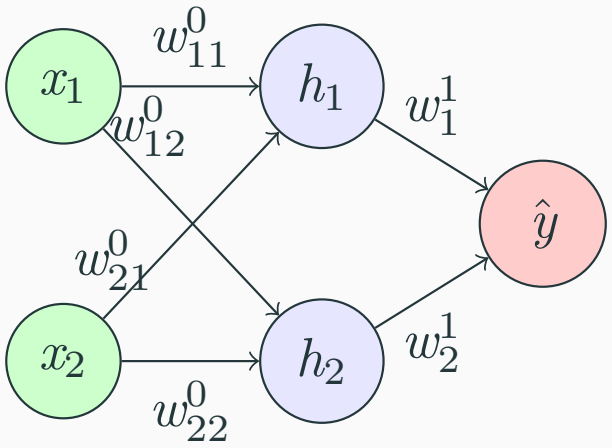


In red the estimation of the mean $G_\mu(x, \boldsymbol{\theta}_\mu)$

In green the confidence interval $G_\mu(x, \boldsymbol{\theta}_\mu) \pm \sigma(x_k, \boldsymbol{\theta}_\sigma)$

Gradient backpropagation

Training a neural-net: gradient backpropagation

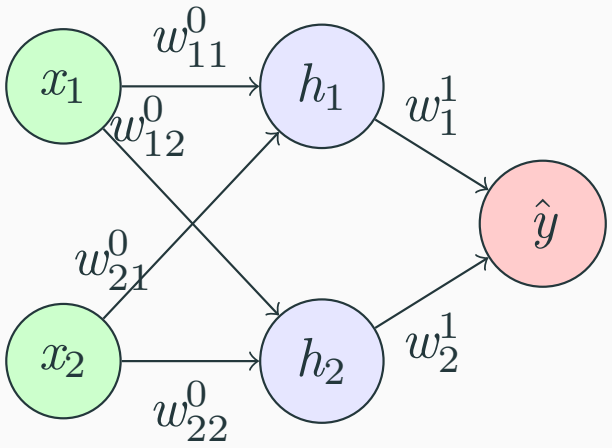


1. Given a couple (x, y)

Objective

Determination of the best set of weights $\mathbf{w}$ to minimize the Loss function $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2$.
Calculation of $\partial L / \partial w$

Training a neural-net: gradient backpropagation

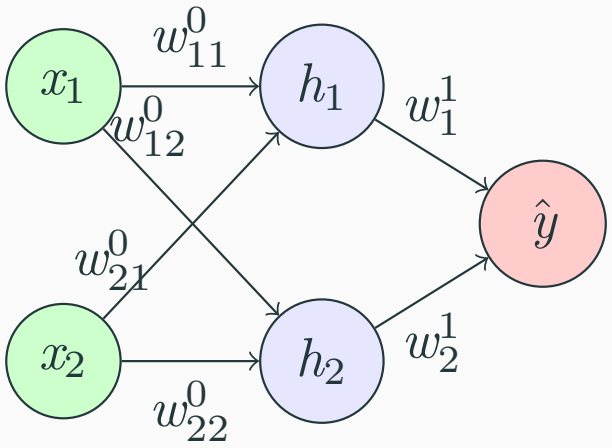


1. Given a couple (x, y)
2. Forward computation:
   $h_j = f_0(\sum_{i=1}^{2} w^0_{ij}.x_i)$
   $\hat{y} = f_1(\sum_{j=1}^{2} w^1_j.h_j)$

Objective

Determination of the best set of weights $\mathbf{w}$ to minimize the Loss function $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2$.
Calculation of $\partial L/\partial w$

Training a neural-net: gradient backpropagation

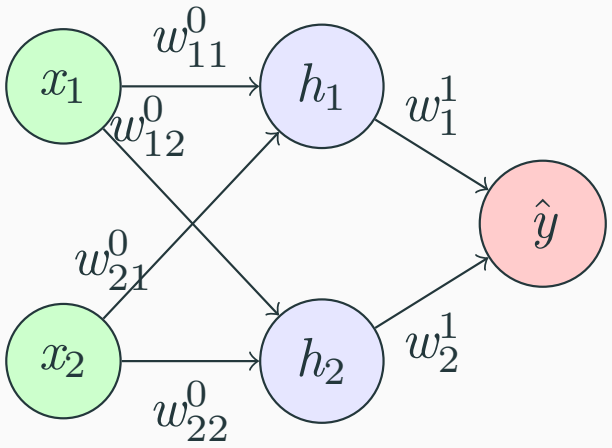


1. Given a couple (x, y)
2. Forward computation:
   $h_j = f_0(\sum_{i=1}^{2} w^0_{ij}.x_i)$
   $\hat{y} = f_1(\sum_{j=1}^{2} w^1_j.h_j)$
3. Compute the gradient of the loss:
   $\boxed{\partial L/\partial \hat{y}}$

Objective

Determination of the best set of weights $\mathbf{w}$ to minimize the Loss function $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2$.
Calculation of $\partial L/\partial w$

Training a neural-net: gradient backpropagation

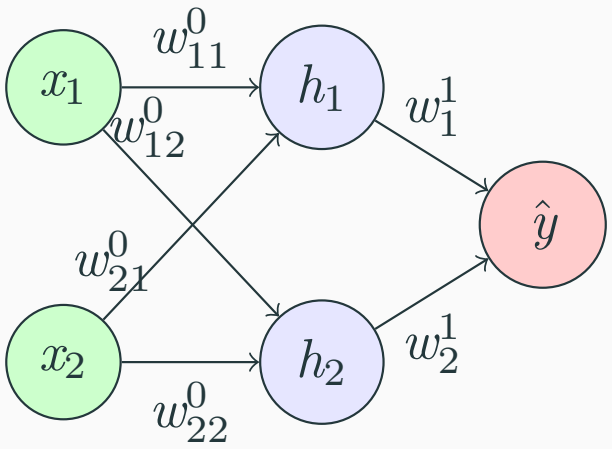


Objective

Determination of the best set of weights $\mathbf{w}$ to minimize the Loss function $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2$.
Calculation of $\partial L/\partial w$

1. Given a couple (x, y)
2. Forward computation:
   $h_j = f_0(\sum_{i=1}^{2} w^0_{ij}.x_i)$
   $\hat{y} = f_1(\sum_{j=1}^{2} w^1_j.h_j)$
3. Compute the gradient of the loss:
   $\boxed{\partial L/\partial \hat{y}}$
4. Gradient Backpropagation:

Training a neural-net: gradient backpropagation

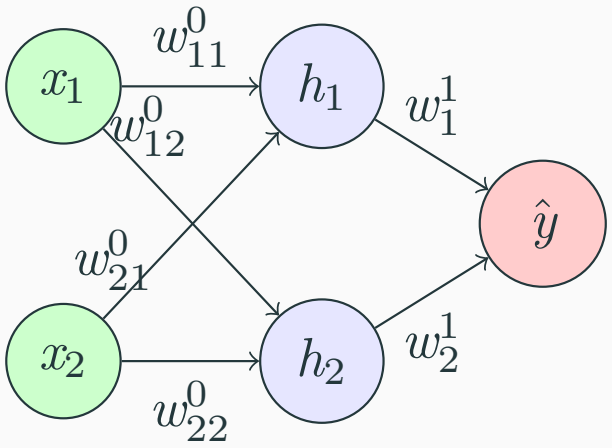


Objective

Determination of the best set of weights $\mathbf{w}$ to minimize the Loss function $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2$.
Calculation of $\partial L/\partial w$

1. Given a couple (x, y)
2. Forward computation:
   $h_j = f_0(\sum_{i=1}^{2} w^0_{ij}.x_i)$
   $\hat{y} = f_1(\sum_{j=1}^{2} w^1_j.h_j)$
3. Compute the gradient of the loss:
   $\boxed{\partial L/\partial \hat{y}}$
4. Gradient Backpropagation:
   - Layer 1
     $\partial L/\partial w^1_j = \boxed{\partial L/\partial \hat{y}}.\partial f_1/\partial w^1_j$
     $\boxed{\partial L/\partial h_j} = \boxed{\partial L/\partial \hat{y}}.\partial f_1/\partial h_j$

Training a neural-net: gradient backpropagation



Objective

Determination of the best set of weights $\mathbf{w}$ to minimize the Loss function $L(\mathbf{w}) = ||\hat{y}(\mathbf{w}) - y||^2$.
Calculation of $\partial L/\partial w$

1. Given a couple (x, y)
2. Forward computation:
   $h_j = f_0(\sum_{i=1}^{2} w^0_{ij}.x_i)$
   $\hat{y} = f_1(\sum_{j=1}^{2} w^1_j.h_j)$
3. Compute the gradient of the loss:
   $\boxed{\partial L/\partial \hat{y}}$
4. Gradient Backpropagation:
   - Layer 1
     $\partial L/\partial w^1_j = \boxed{\partial L/\partial \hat{y}}.\partial f_1/\partial w^1_j$
     $\boxed{\partial L/\partial h_j} = \boxed{\partial L/\partial \hat{y}}.\partial f_1/\partial h_j$
   - Layer 0
     $\partial L/\partial w^0_{ij} = \boxed{\partial L/\partial h_j}.\partial f_1/\partial w^0_{ij}$

Optimizing a machine learning (gradient method)

Optimizing the loss

Several loss function (depending on the problem) can be defined.

For example, Mean Square Error:

Method

Find a minimum of L by adjusting the parameters (weights) $\mathbf{w}$ given the gradient of the loss with respect to the weights $\nabla_{\mathbf{w}} L$.

Batch Vs Stochastic training

Dataset: (X, Y) with N samples denoted $(\mathbf{x_i}, y_i)$

Batch gradient:

Require: Learning rate(s): ν_k
Require: Initial weights: $\mathbf{w}$
$k \leftarrow 1$
while stopping criterion not met **do**
Compute gradient:
$\mathbf{g} \leftarrow \frac{1}{N} \sum_i^N \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i))$
Update weights: $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$
$k \leftarrow k + 1$
end while

1 Update / N forwards

Batch Vs Stochastic training

Dataset: (X, Y) with N samples denoted $(\mathbf{x_i}, y_i)$

Batch gradient:

Require: Learning rate(s): ν_k
Require: Initial weights: $\mathbf{w}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Compute gradient:
  $\mathbf{g} \leftarrow \frac{1}{N} \sum_i^N \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i))$
    Update weights: $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$
    $k \leftarrow k + 1$
  end while

1 Update / N forwards

Stochastic gradient:

Require: Learning rate(s): ν_k
Require: Initial weights: $\mathbf{w}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Sample an example $(\mathbf{x}, y)$ from (X, Y)
    Compute gradient: $\mathbf{g} \leftarrow \nabla_{\mathbf{w}} L(f(\mathbf{x}, y))$
    Update weights: $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$
    $k \leftarrow k + 1$
  end while

1 Update / 1 forward

Mini-Batch training

Dataset: (X, y) with N samples

Mini-Batch gradient:

Require: Learning rate(s): ν_k
Require: Initial weights: $\mathbf{w}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Sample m examples $(\mathbf{x}_i, y_i)$ from (X, y)
    Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \sum_i^m \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i)$
    Update weights: $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$
    $k \leftarrow k + 1$
  end while

Mini-Batch training

Dataset: (X, y) with N samples

Mini-Batch gradient:

Require: Learning rate(s): ν_k
Require: Initial weights: $\mathbf{w}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Sample m examples $(\mathbf{x}_i, y_i)$ from (X, y)
    Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \sum_i^m \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, y_i)$
    Update weights: $\mathbf{w} \leftarrow \mathbf{w} - \nu_k \mathbf{g}$
    $k \leftarrow k + 1$
  end while

1 Update / m forward

$m = 1$: Pure stochastic gradient.

$m = N$: Batch gradient

Convolutional Neural Networks

Convolutional neural net

X: an image

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}
x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}

$$\begin{matrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{matrix}$$

$\mathbf{w}$

h: first feature

h_{11}	h_{12}	h_{13}	h_{14}
h_{21}	h_{22}	h_{23}	h_{24}
h_{31}	h_{32}	h_{33}	h_{34}
h_{41}	h_{42}	h_{43}	h_{44}

Perform a standard convolution

$$h_{i,j} = \sum_{k=1}^{3} \sum_{l=1}^{3} x_{i+k-1,j+l-1} \cdot w_{k,l}$$

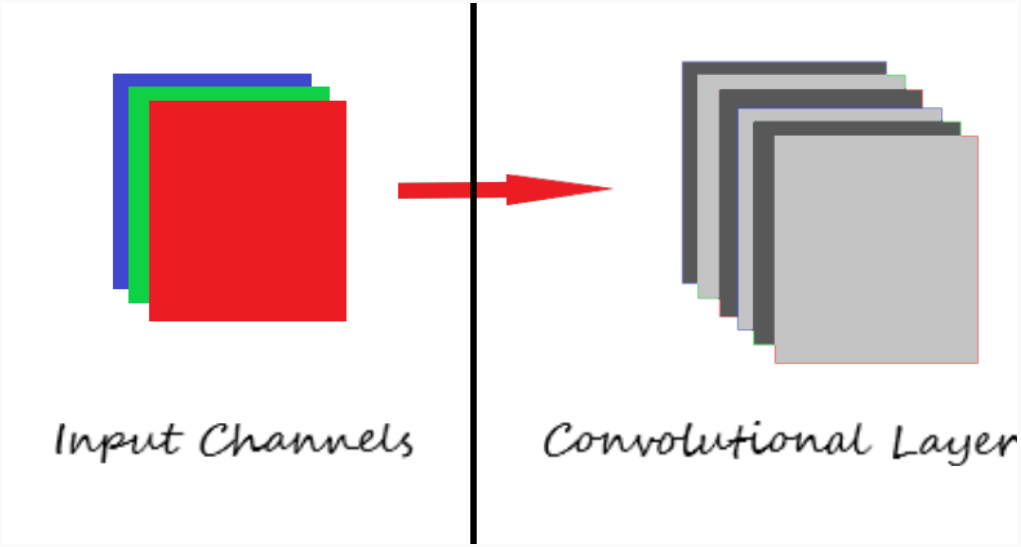
Main parameters of a convolutional layer

- Size of the filter K

Main parameters of a convolutional layer

- Size of the filter K
- Number of filters p

  A convolutional layer is composed of p convolutions (size of layer) extracting p features from the data.

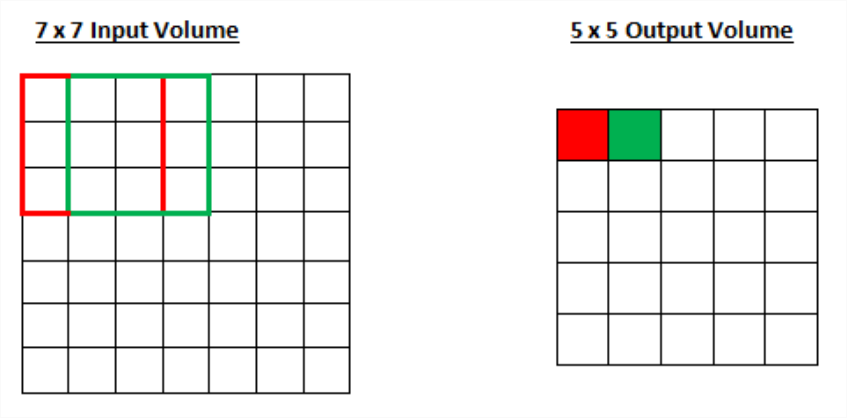


$O = \frac{W-K+2P}{S} + 1$, where O is the output size and W the input size.

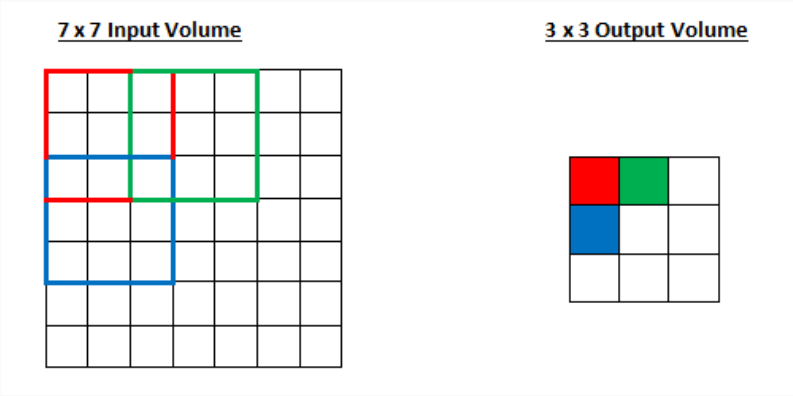
Main parameters of a convolutional layer

- Size of the filter K
- Number of filters p
- Strides S

$S = 1$



$S = 2$

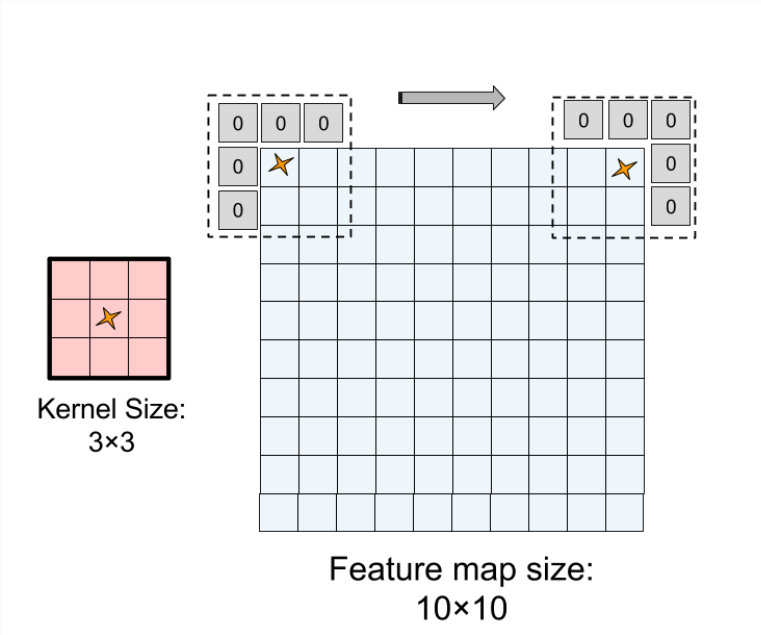


$O = \frac{W-K+2P}{S} + 1$, where O is the output size and W the input size.

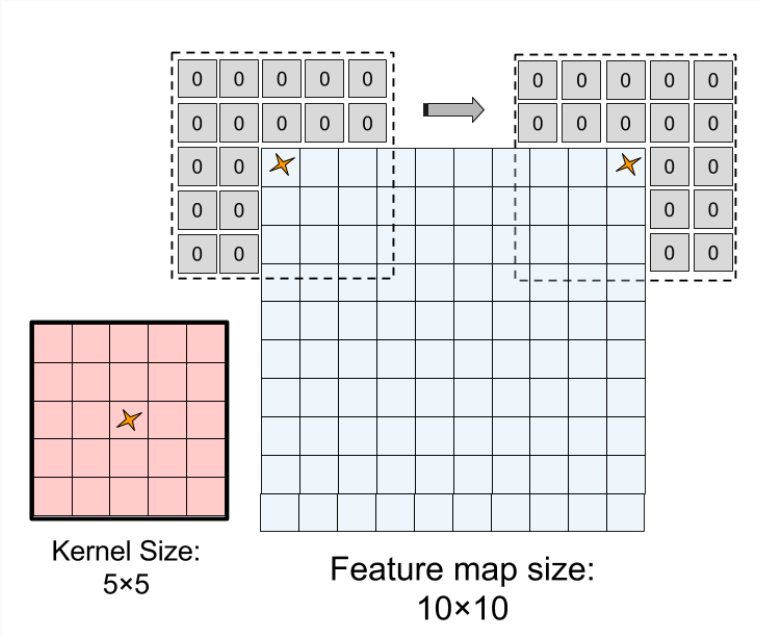
Main parameters of a convolutional layer

- Size of the filter K
- Number of filters p
- Strides S
- Padding P

$P = 1$



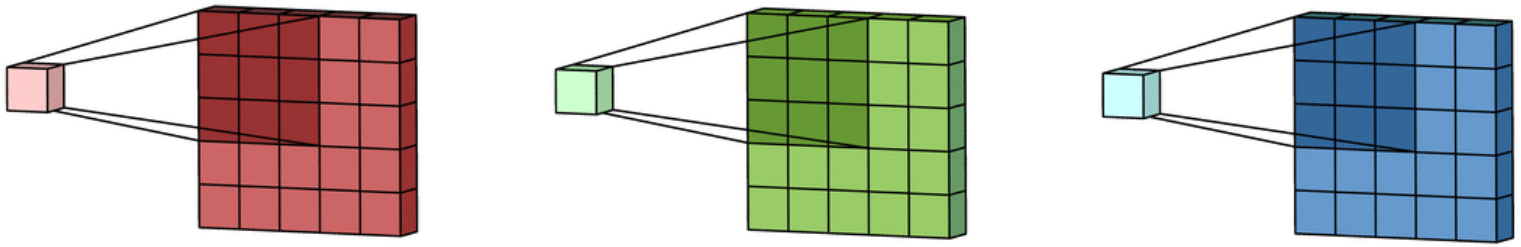
$P = 2$



$O = \frac{W-K+2P}{S} + 1$, where O is the output size and W the input size.

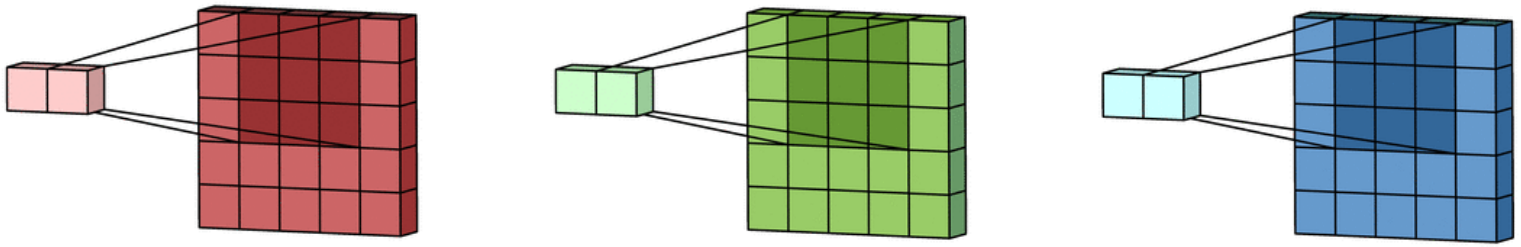
Summary of Convolutional layer steps

1. Convolution



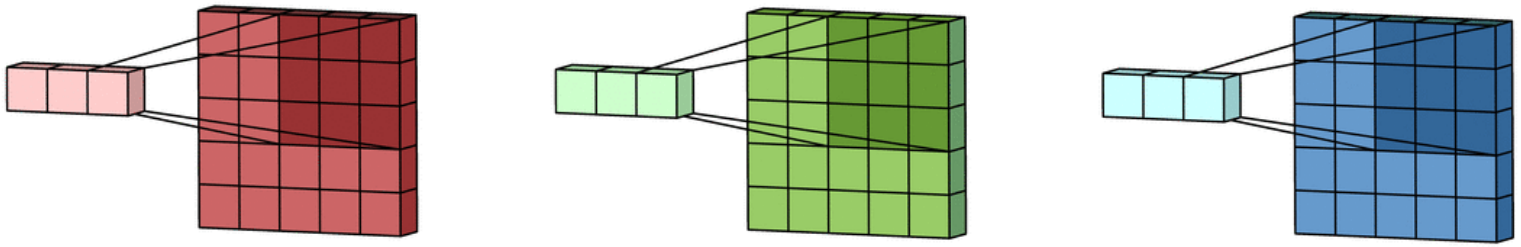
Summary of Convolutional layer steps

1. Convolution



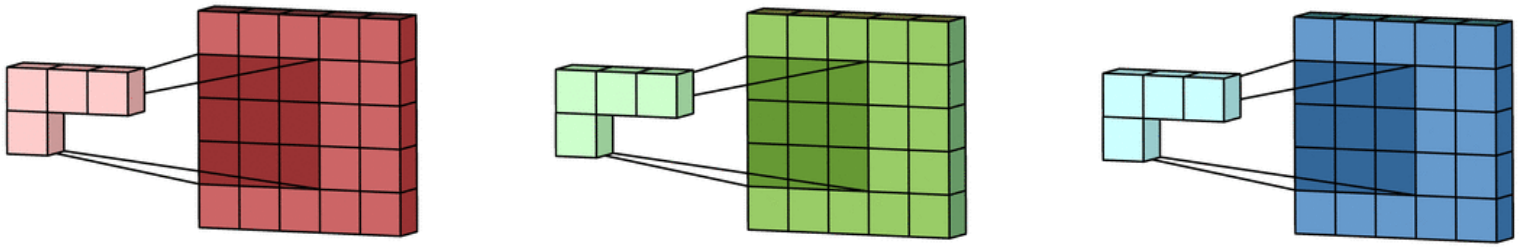
Summary of Convolutional layer steps

1. Convolution



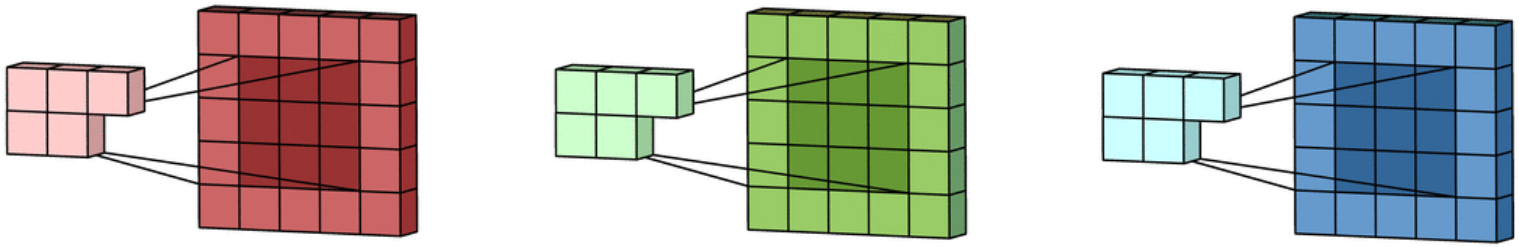
Summary of Convolutional layer steps

1. Convolution



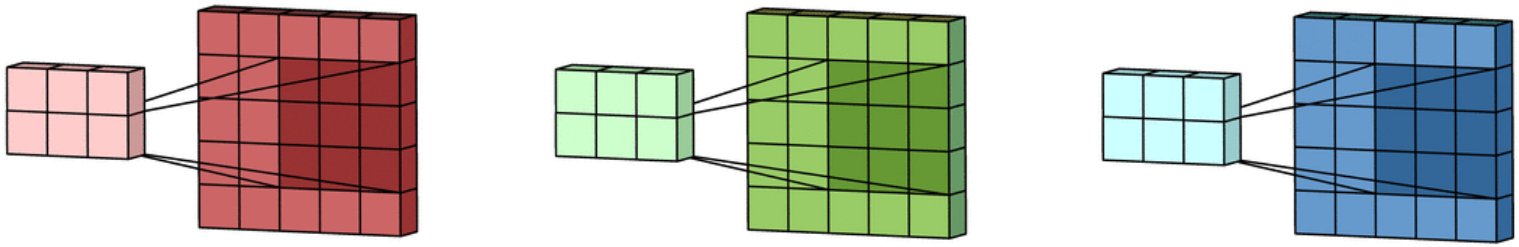
Summary of Convolutional layer steps

1. Convolution



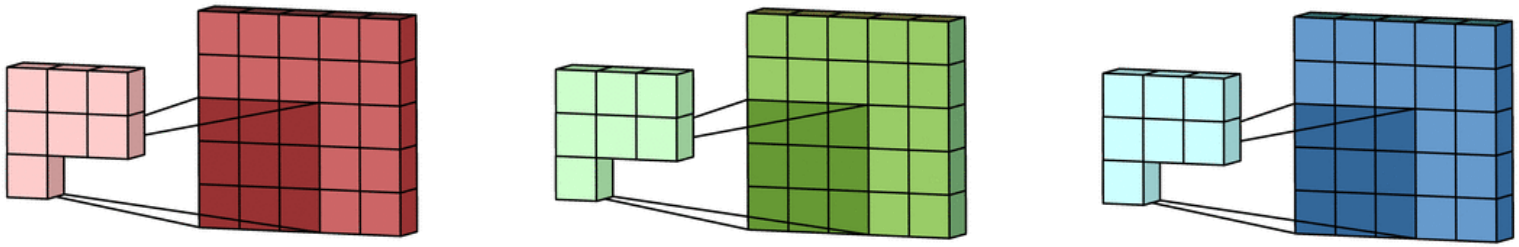
Summary of Convolutional layer steps

1. Convolution



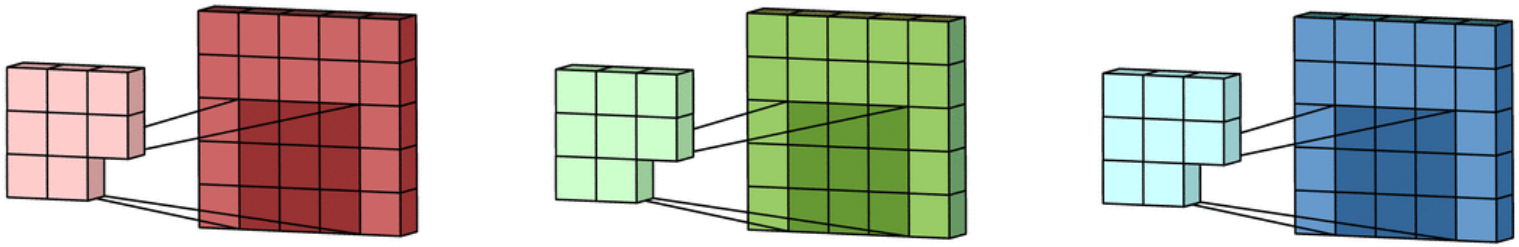
Summary of Convolutional layer steps

1. Convolution



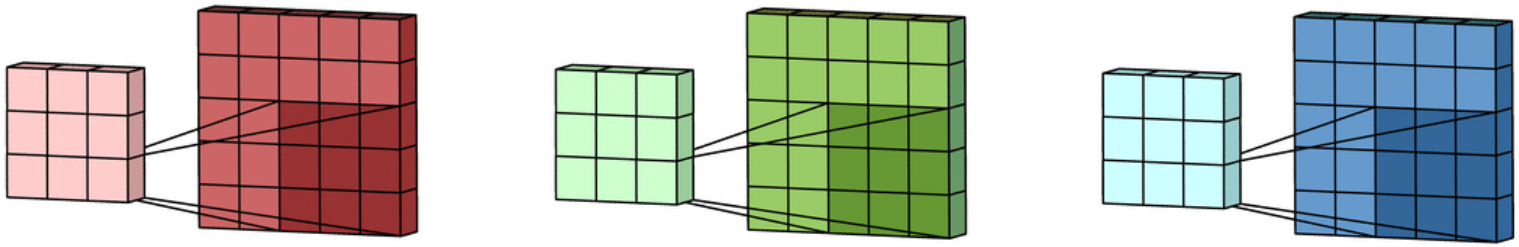
Summary of Convolutional layer steps

1. Convolution



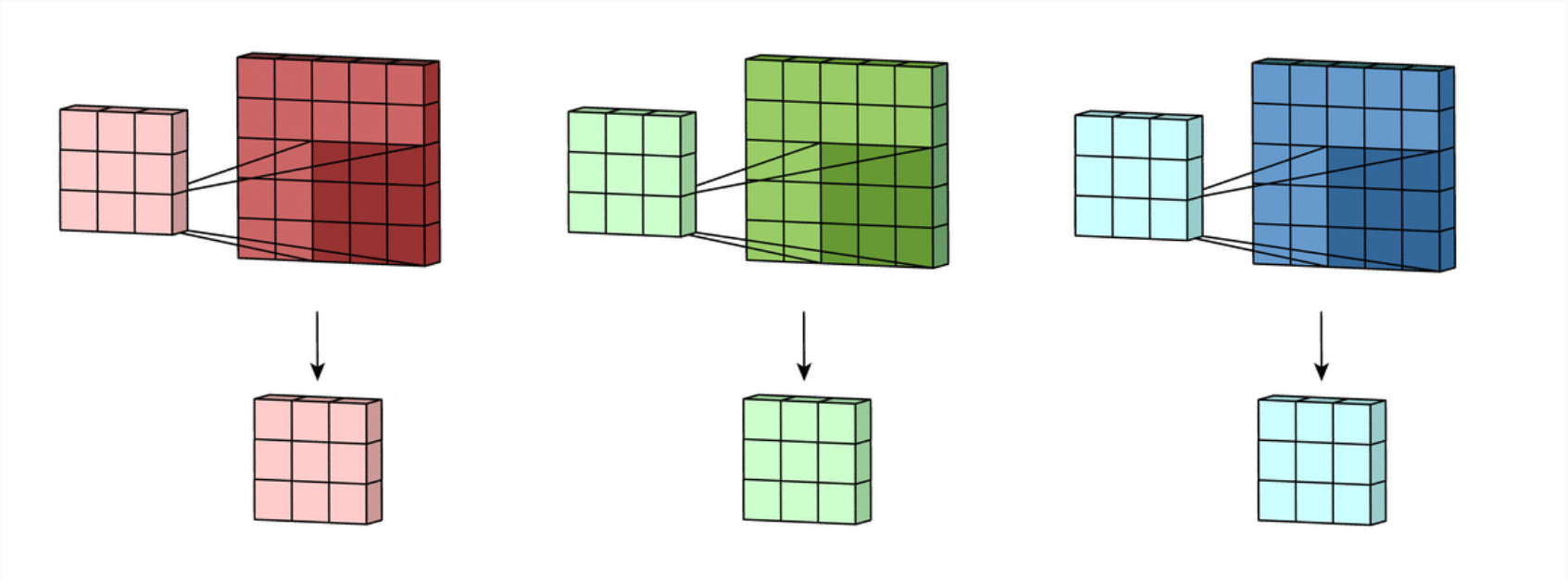
Summary of Convolutional layer steps

1. Convolution



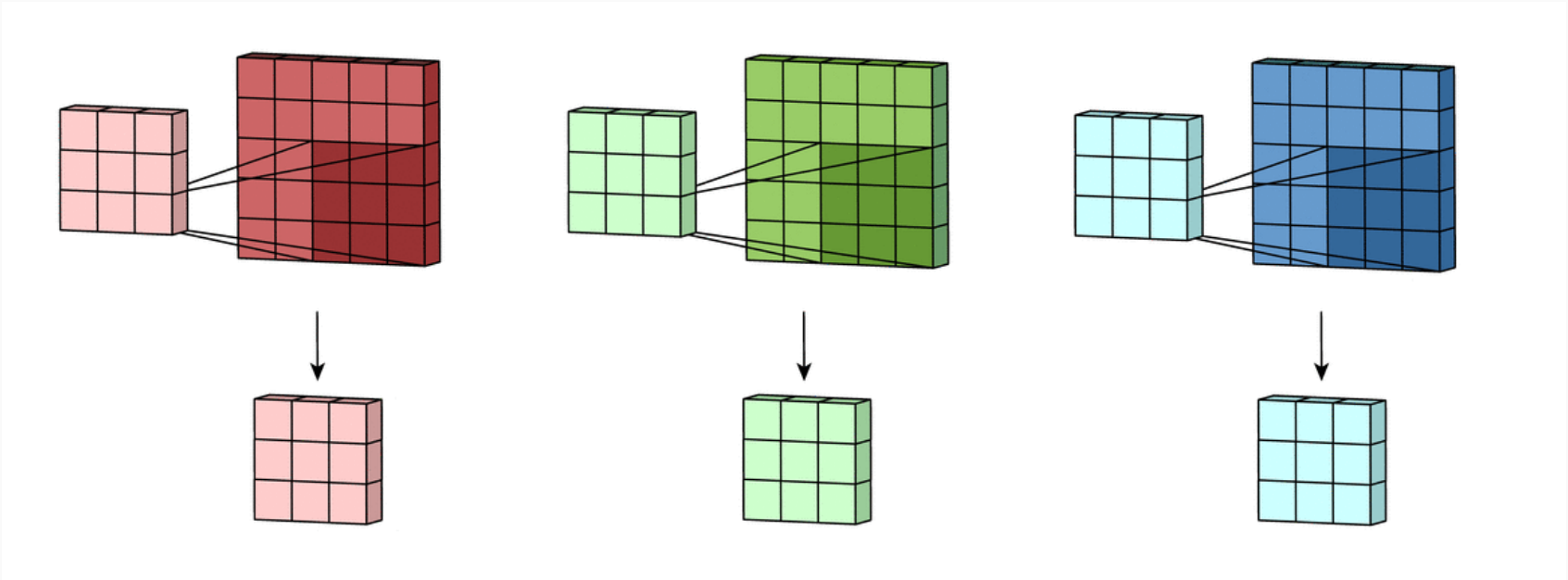
Summary of Convolutional layer steps

1. Convolution



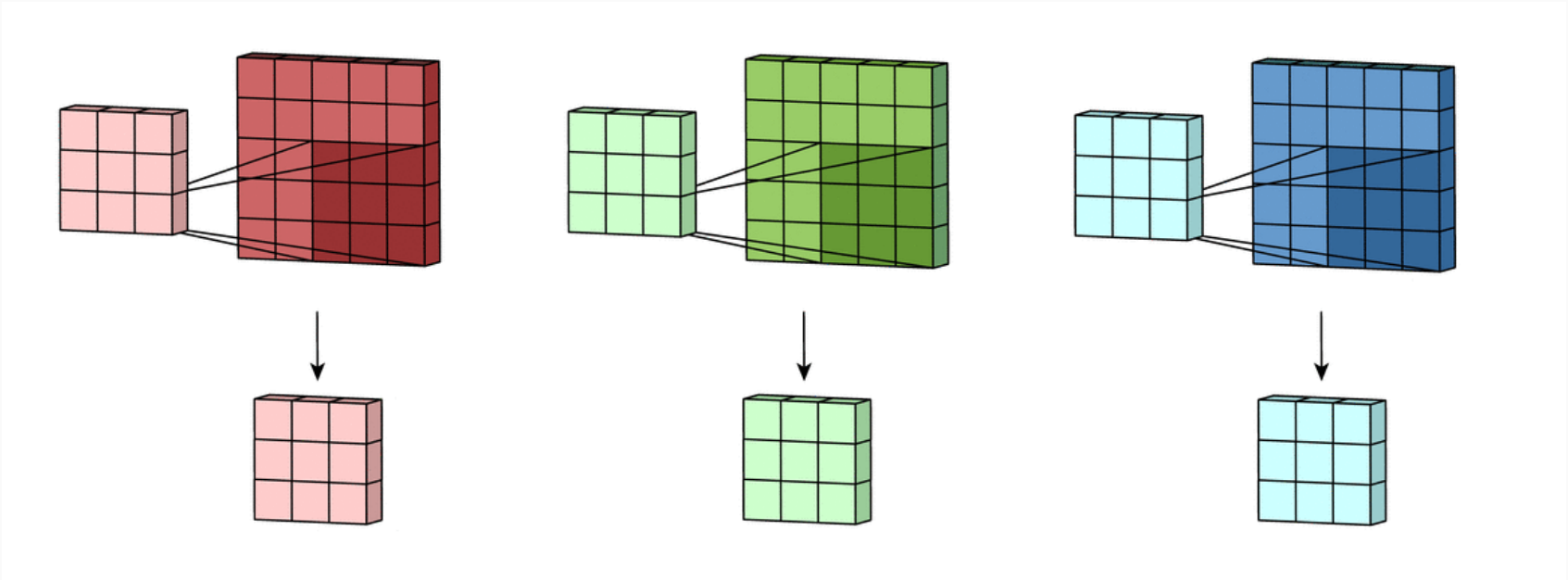
Summary of Convolutional layer steps

1. Convolution

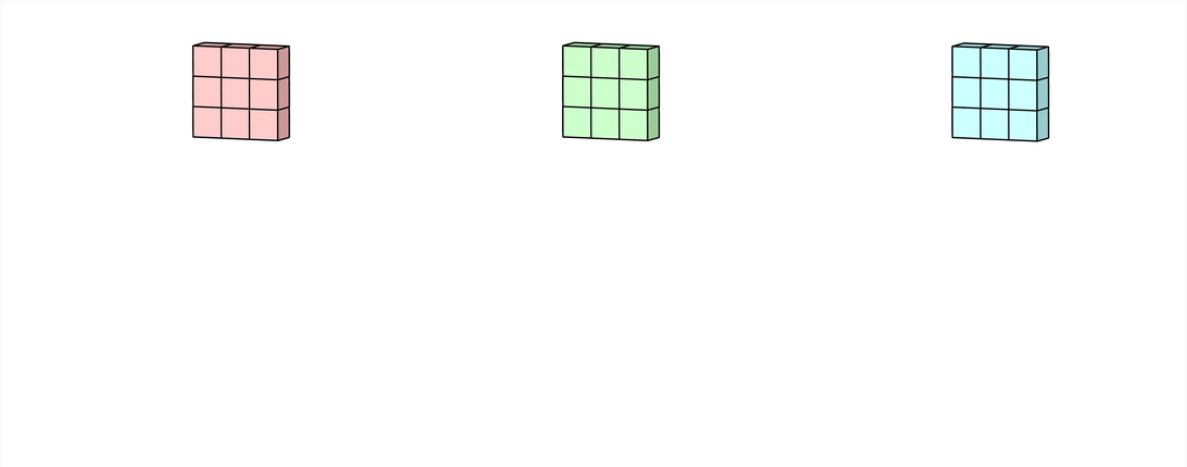


Summary of Convolutional layer steps

1. Convolution

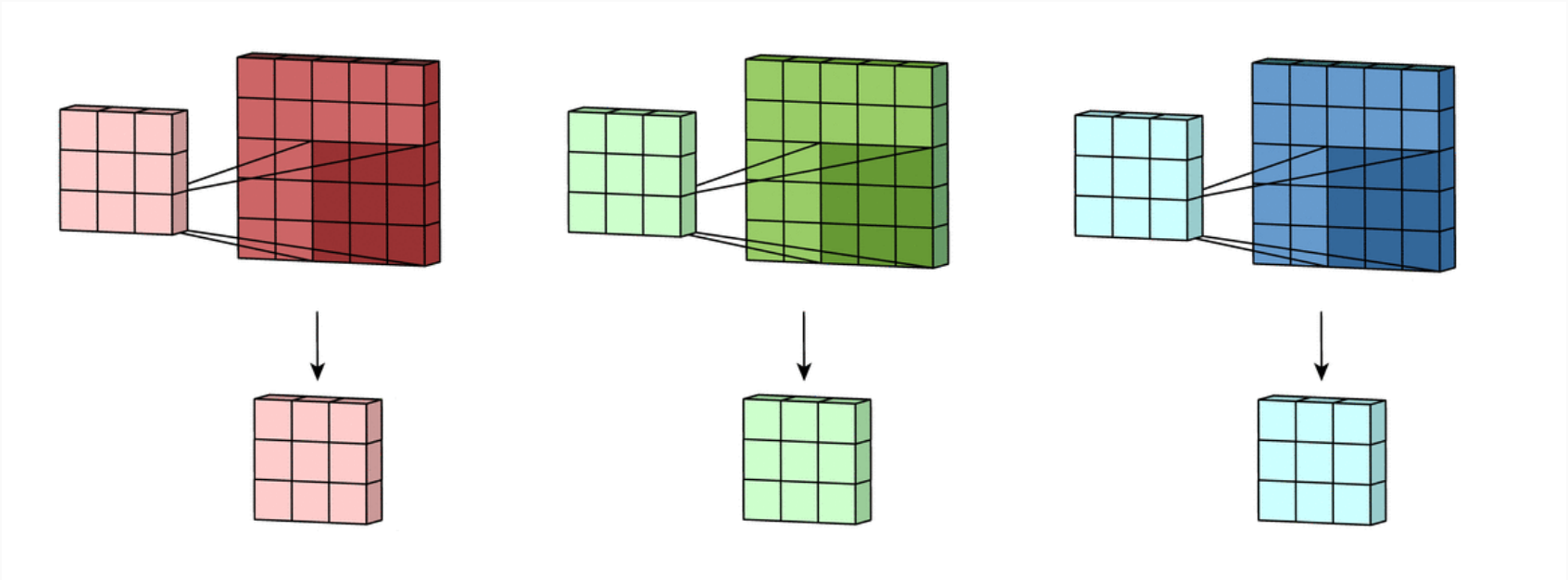


2. Addition

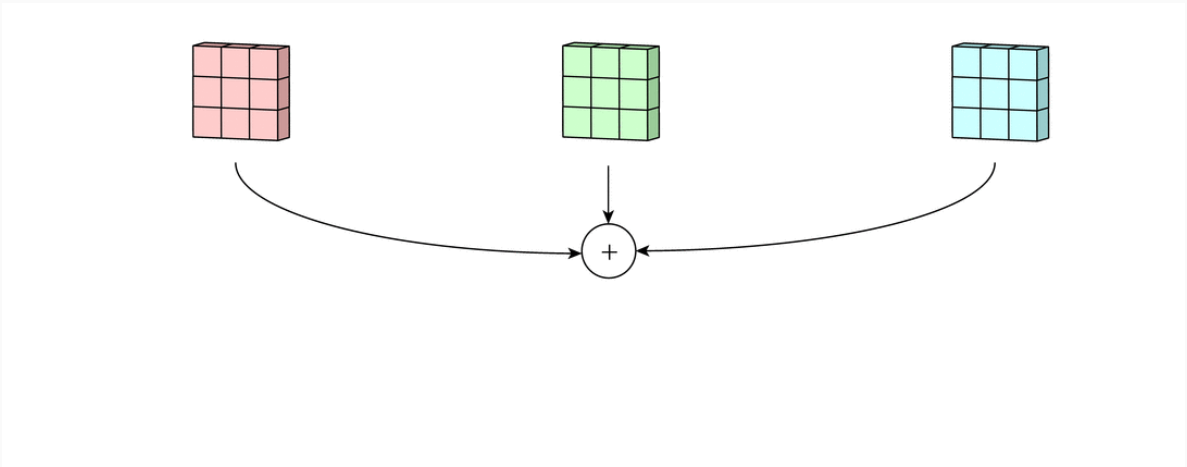


Summary of Convolutional layer steps

1. Convolution

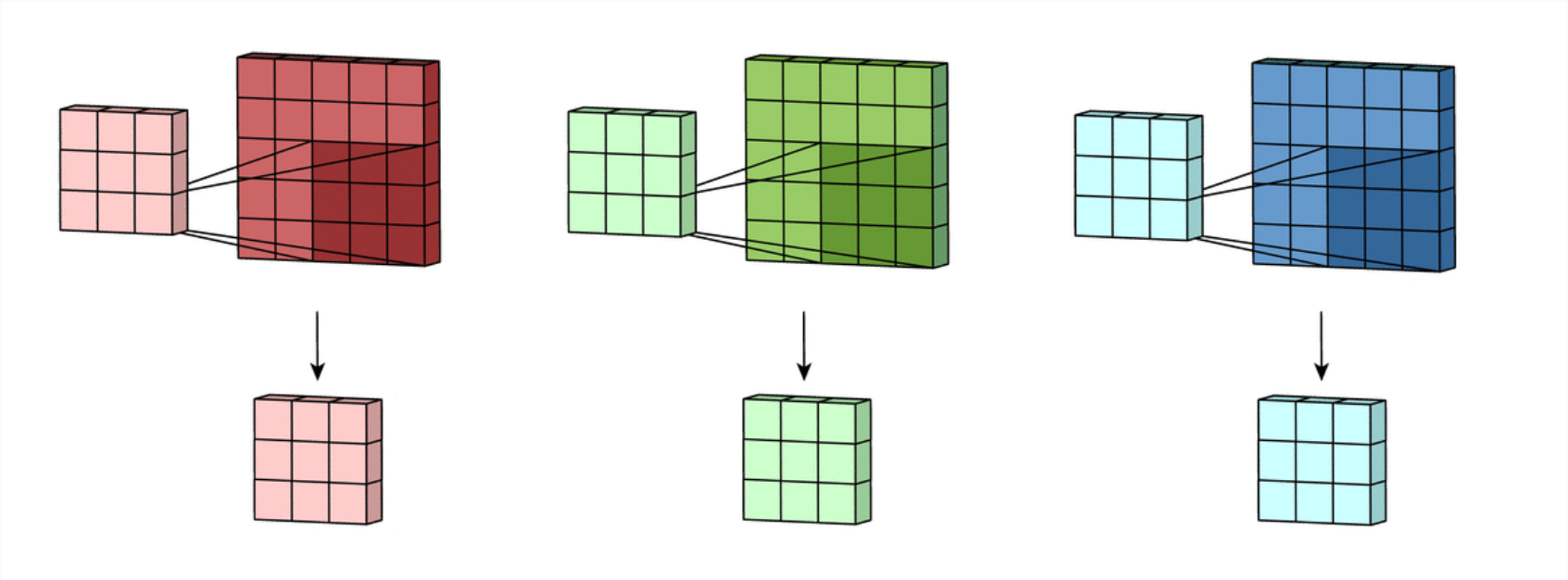


2. Addition

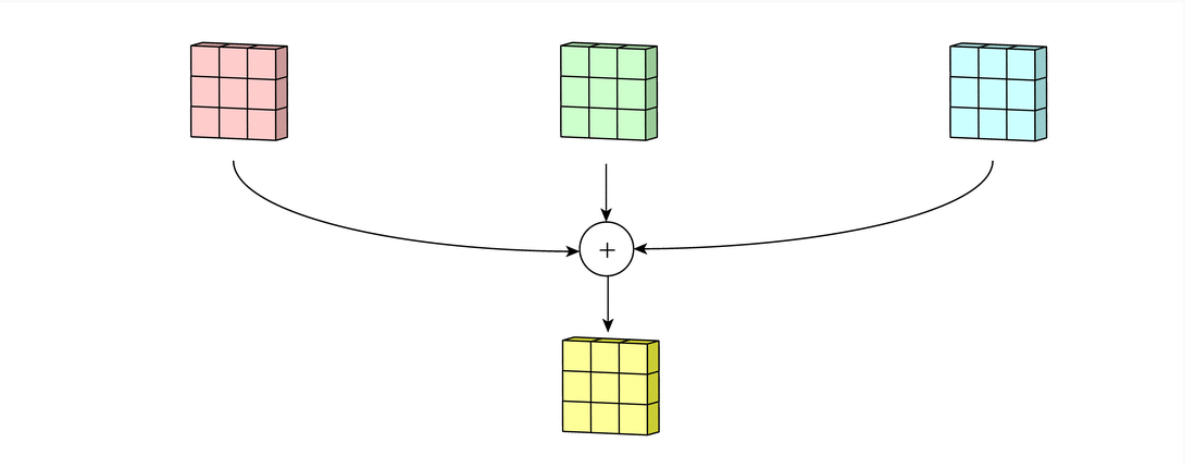


Summary of Convolutional layer steps

1. Convolution

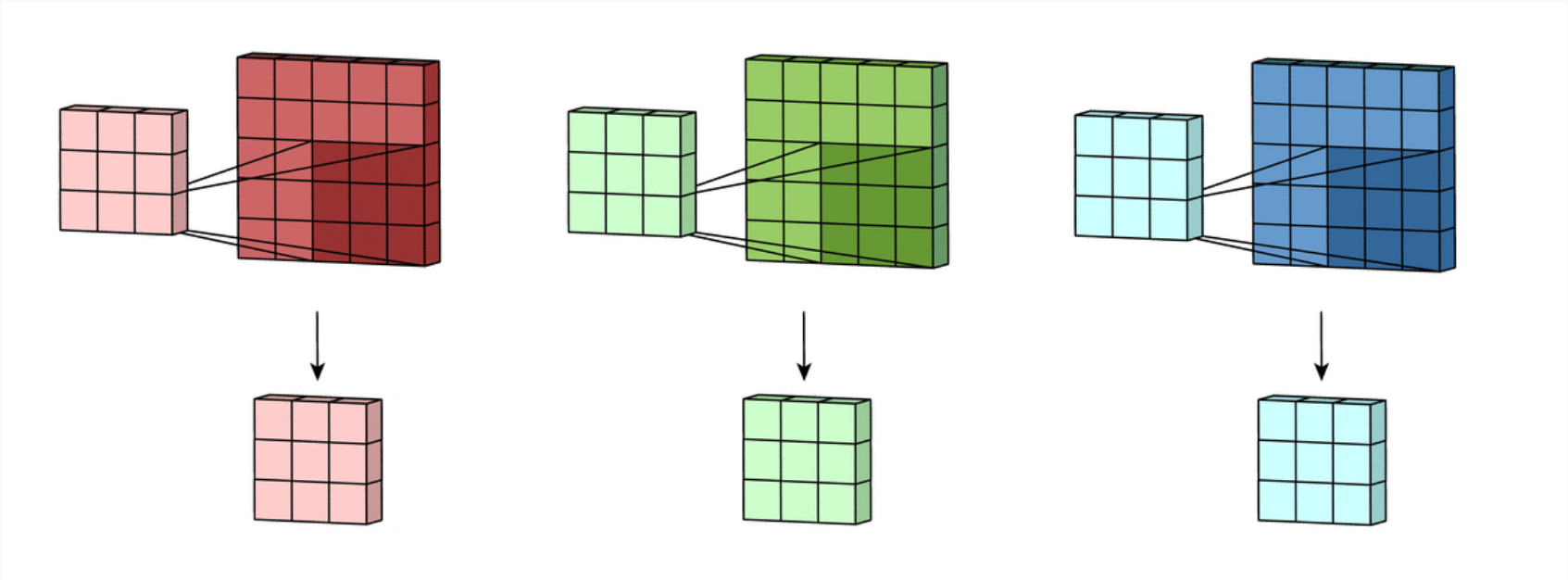


2. Addition

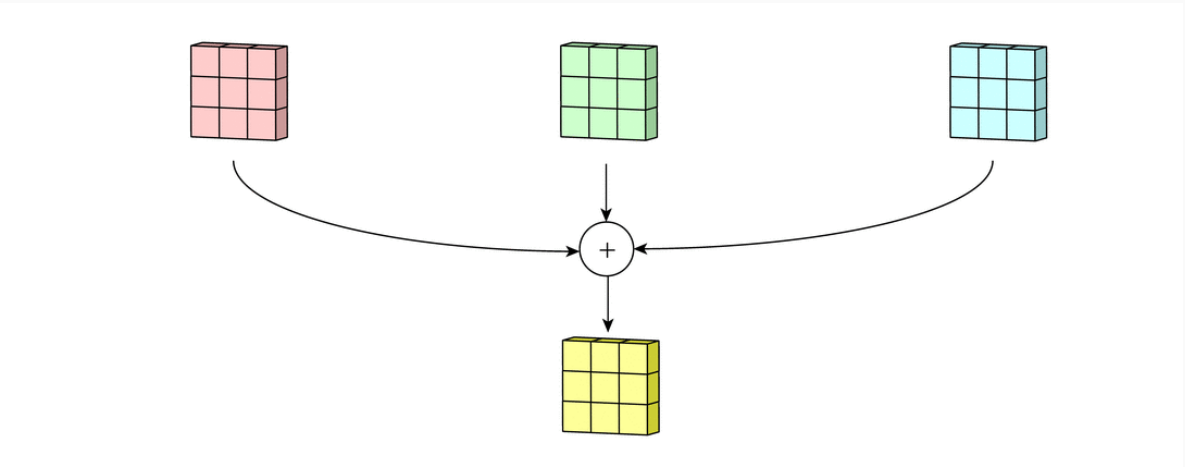


Summary of Convolutional layer steps

1. Convolution

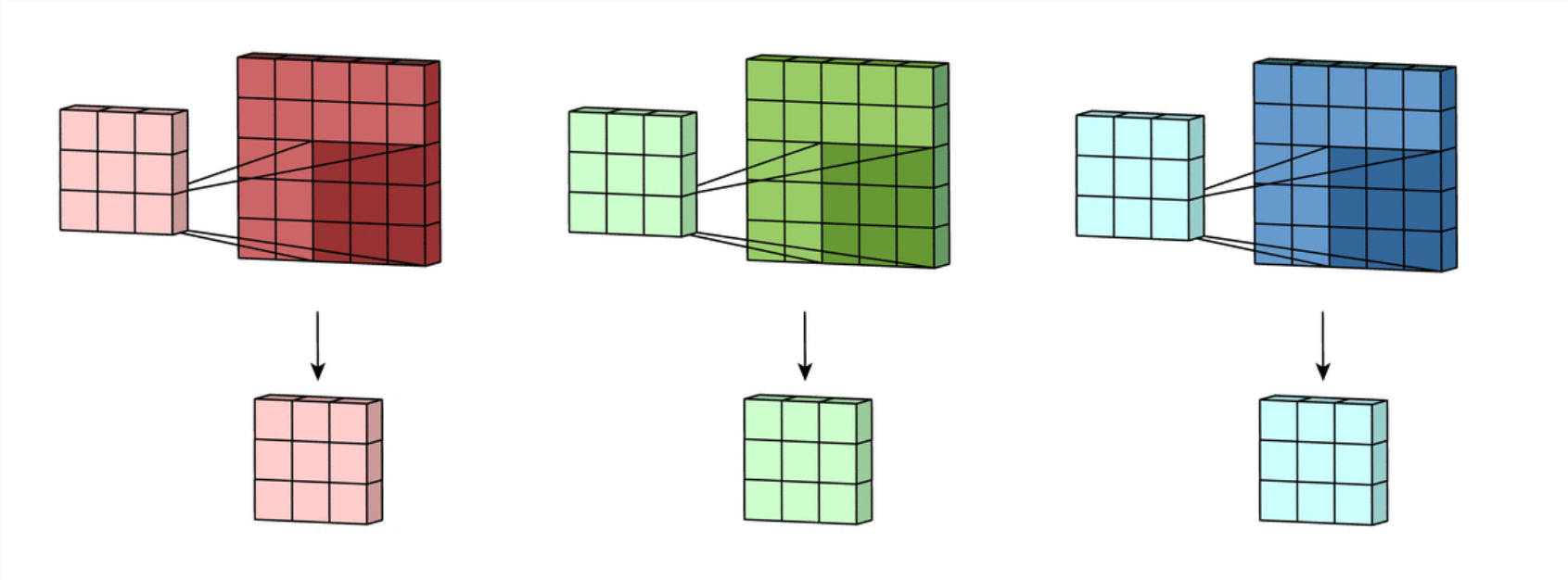


2. Addition

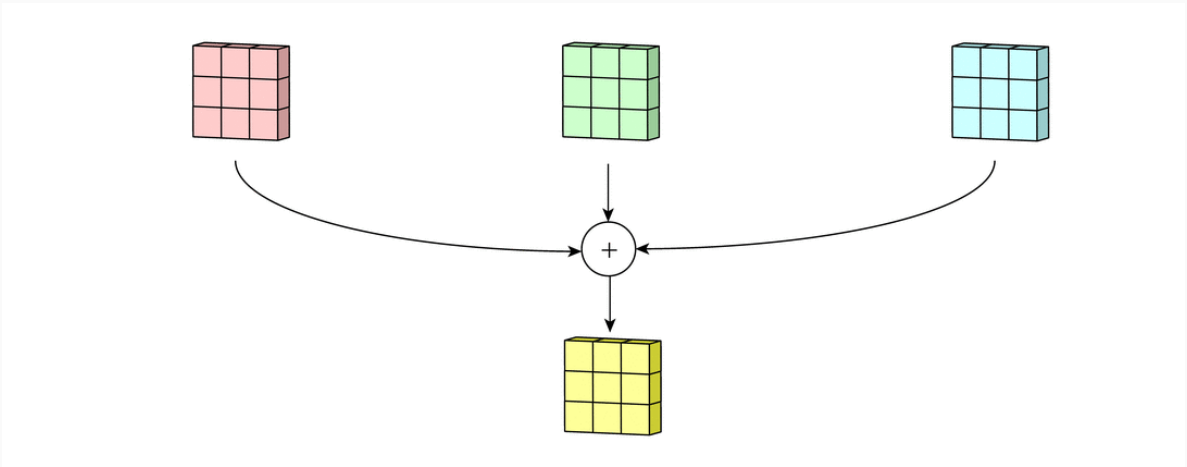


Summary of Convolutional layer steps

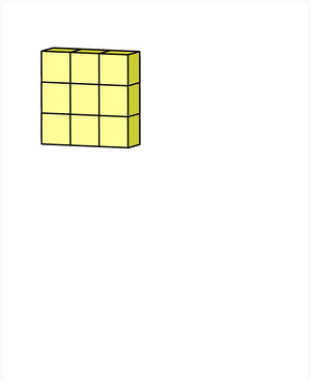
1. Convolution



2. Addition

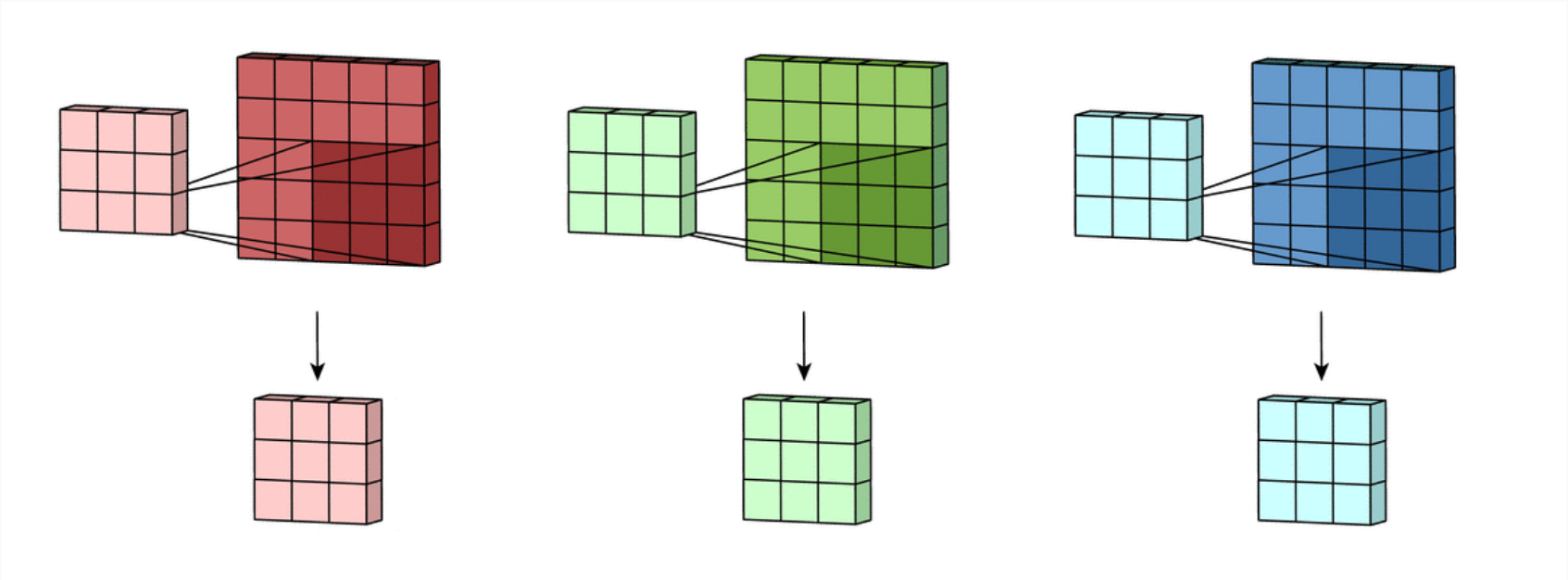


3. Bias

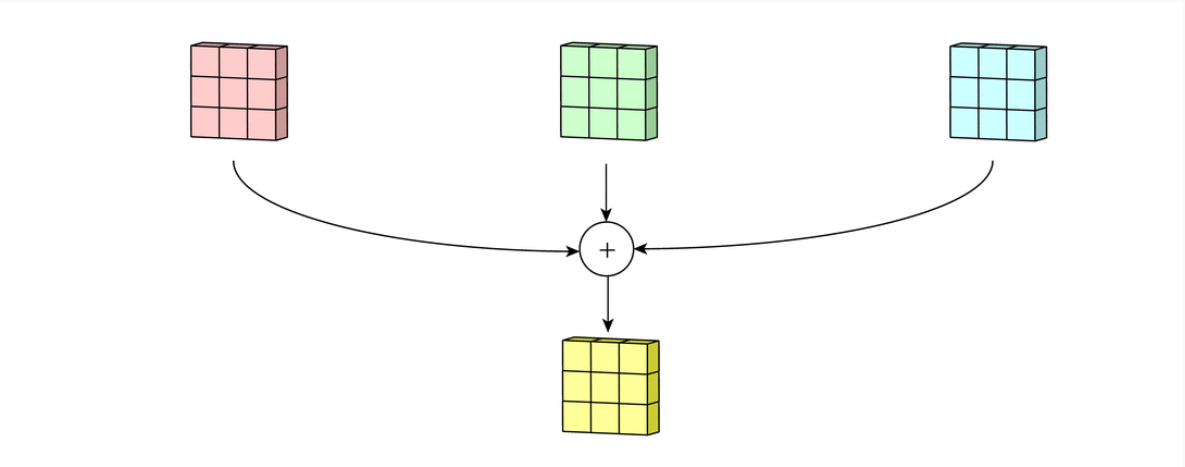


Summary of Convolutional layer steps

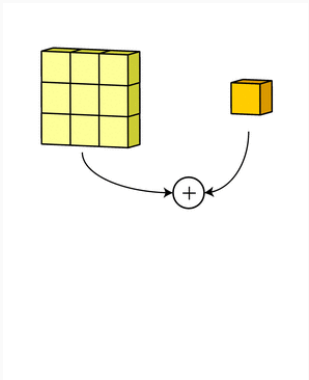
1. Convolution



2. Addition

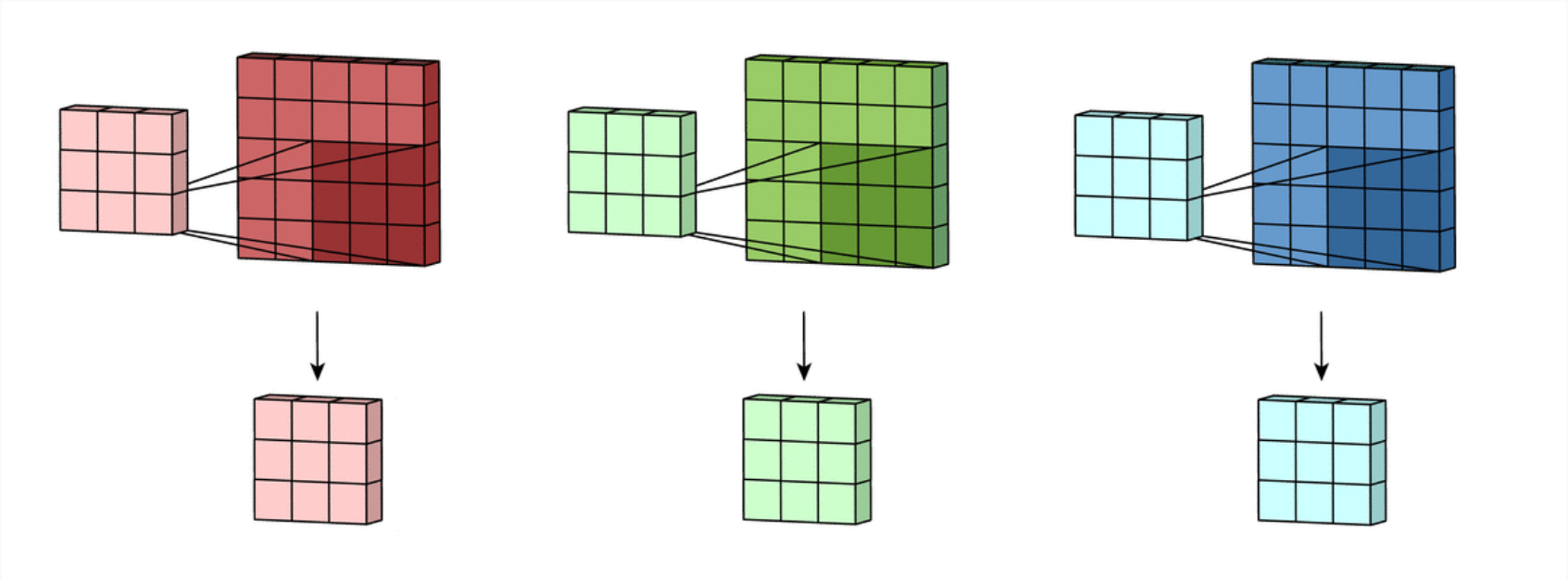


3. Bias

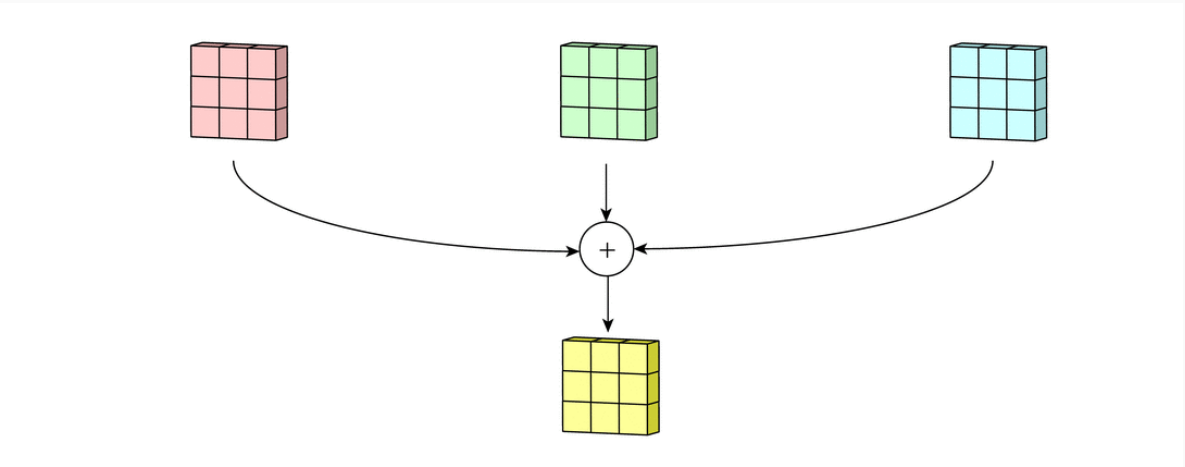


Summary of Convolutional layer steps

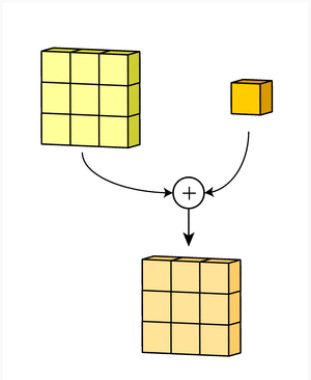
1. Convolution



2. Addition



3. Bias

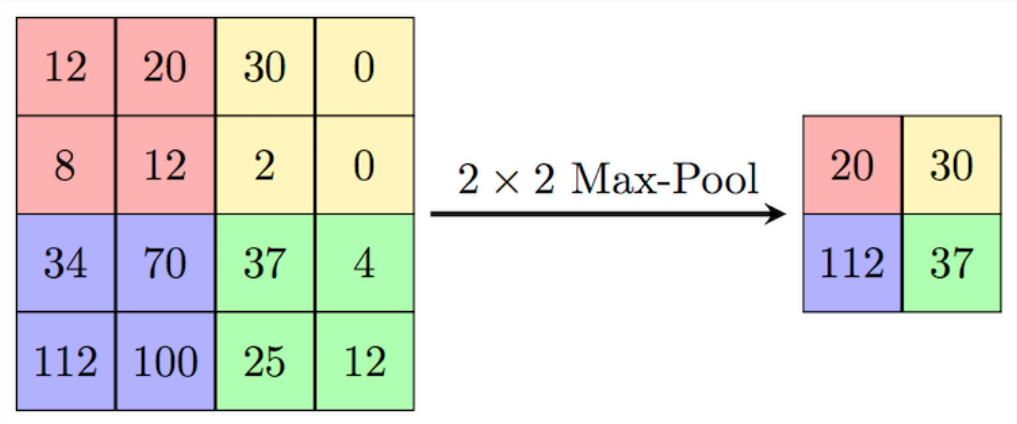


Remarks on Convolutional layers

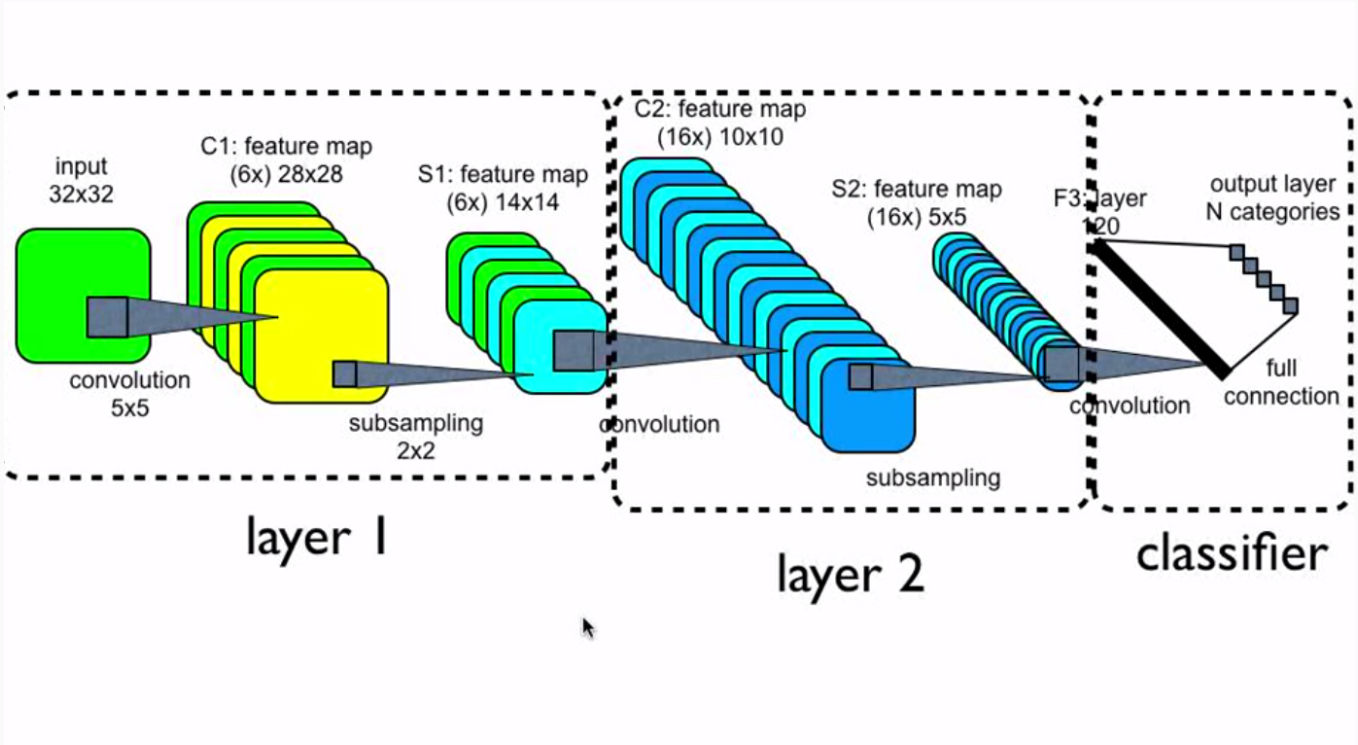
- Convolutional layers are acting locally on the image (But you can still use large scale information by adding more layers)
- Convolutions are invariant by translation (the weights do not depend on the location on the image).
- They can handle images of different sizes.

Max-Pooling

In order to reduce the size of the feature space (en to enhance the gradients), a common operation is to perform a max-pooling.



A traditionnal CNN architecture



Example of AlexNet

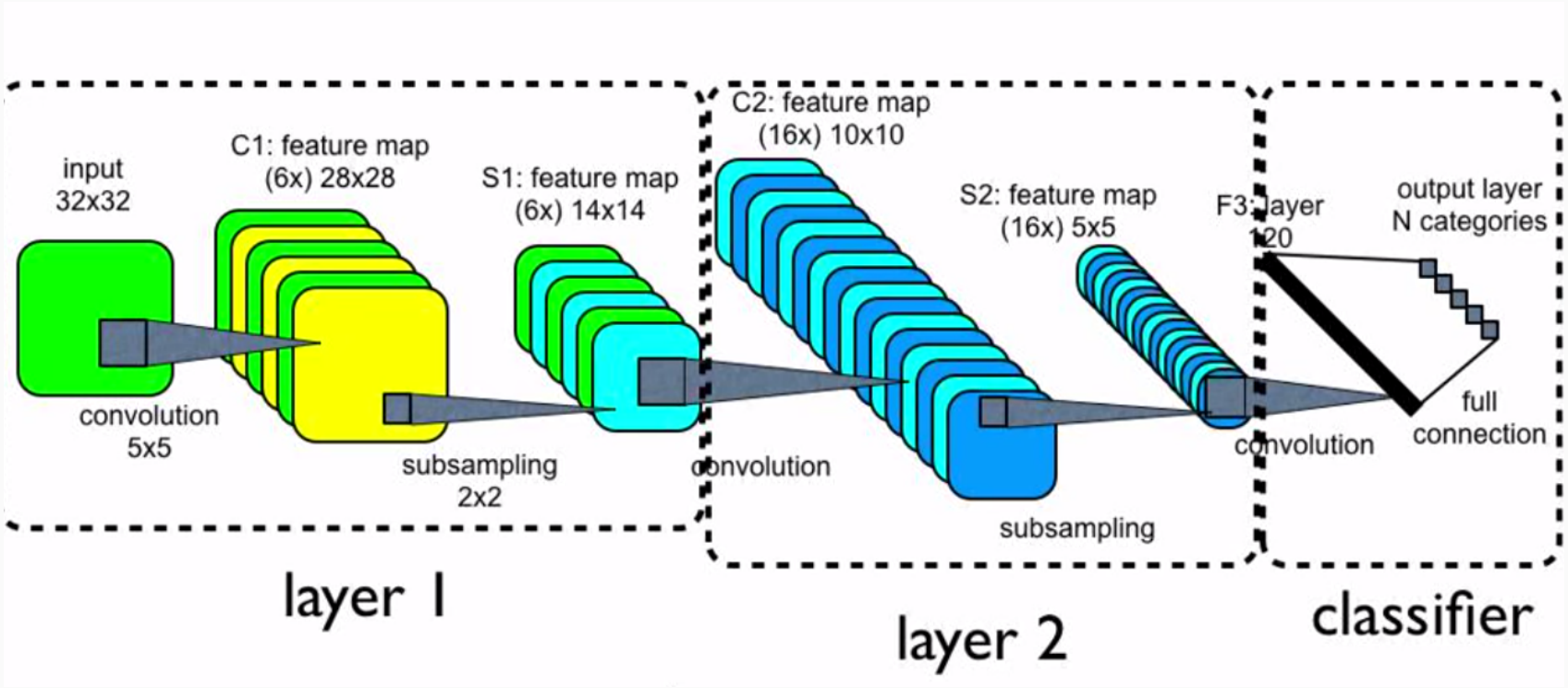
AlexNet is the first Deep architecture used on ImageNet challenge in 2012 and achieved an error of 15.3% (10% better than the previous best classifier). The paper was cited more than 34,000 times.

Alex Krizhevsky and Geoffrey E Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Neural Information Processing Systems (2012), 1–9.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

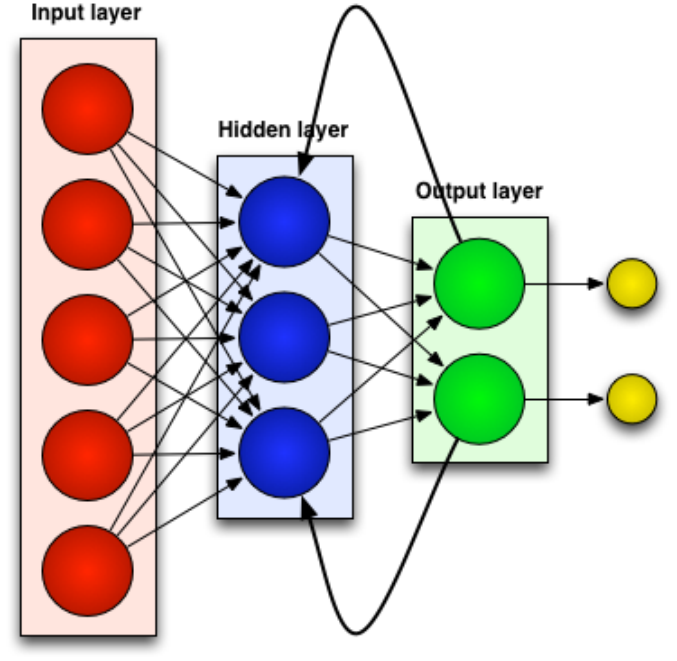
A quick typology of few neural nets

Convolutional neural networks



- Very efficient for computer vision (processing images with spatial patterns)
- Act locally on the input feature space.

Recurrent Neural Networks

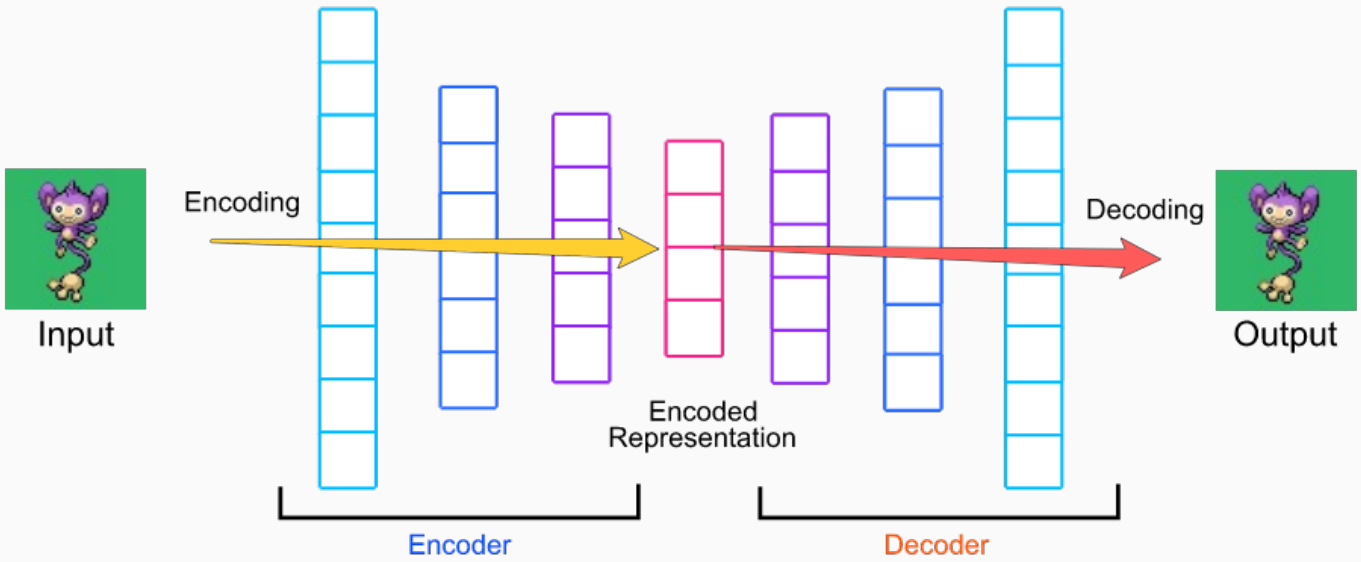


Some popular types of recurrent neural networks:

- Long short-term memory (LSTM)
- Gated Reccurent Unit (GRU)

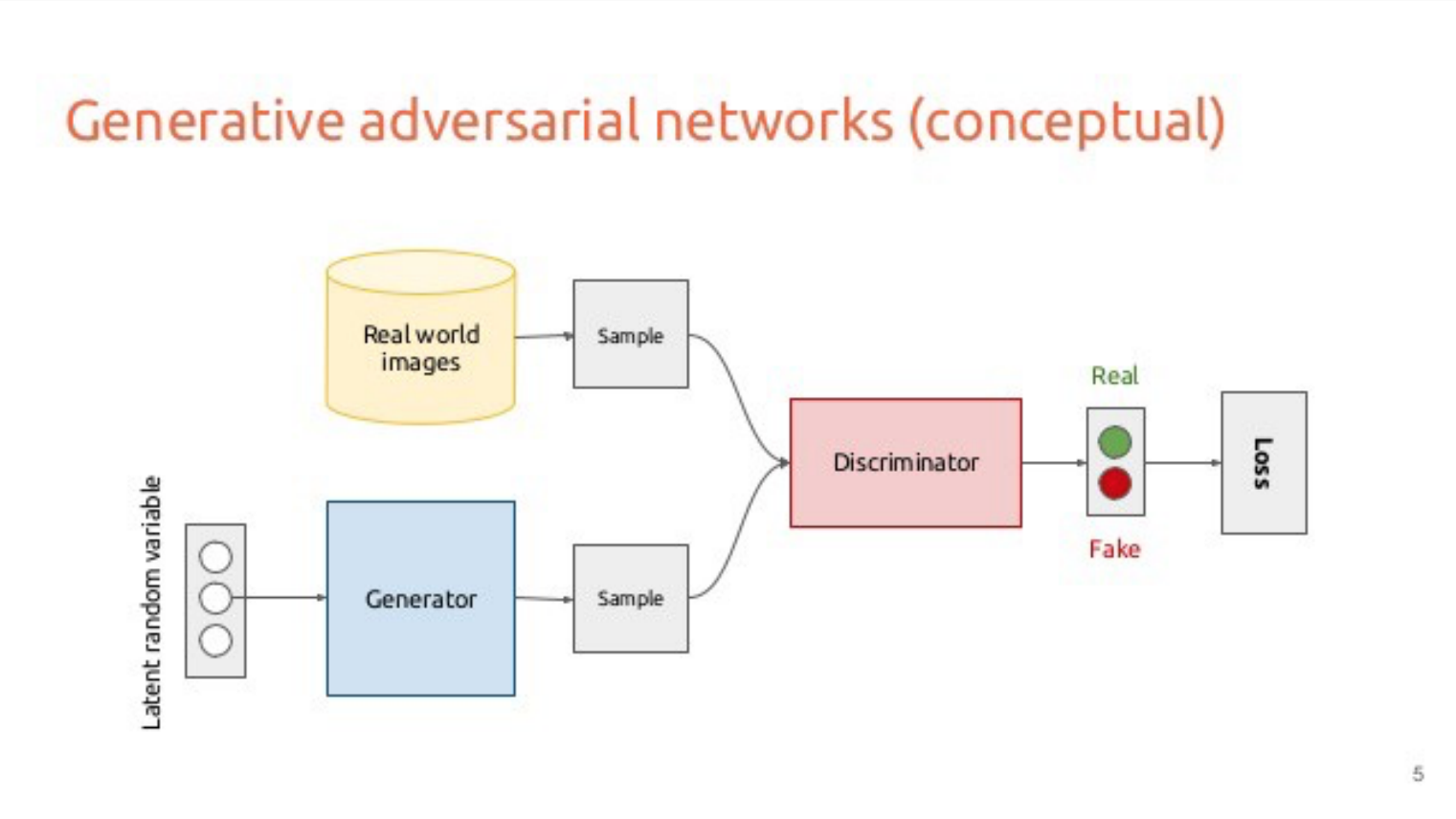
Used in machine translation and text processing

Autoencoders

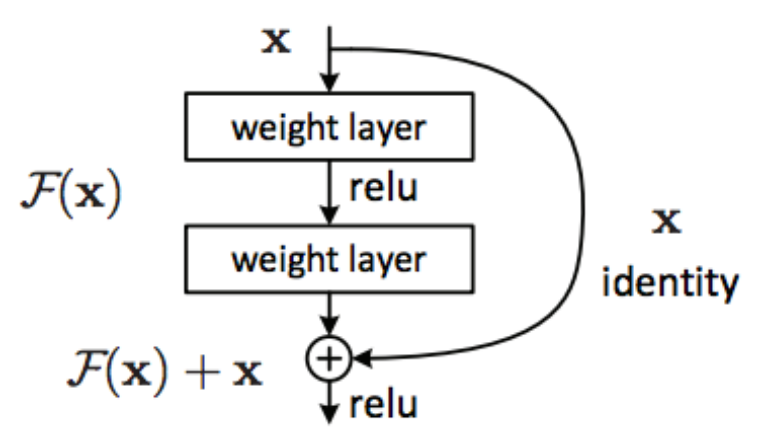


Used in image denoising, compressing, generation,...

Generative adversarial networks



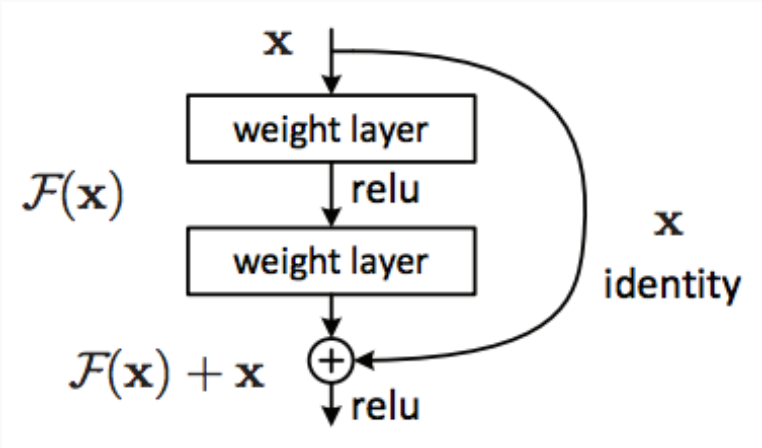
Residual Networks



x : input, y : output

$$y = x + \mathcal{F}(x)$$

Residual Networks



x : input, y : output

$$y = x + \mathcal{F}(x)$$