Machine Learning for biology

V. Monbet

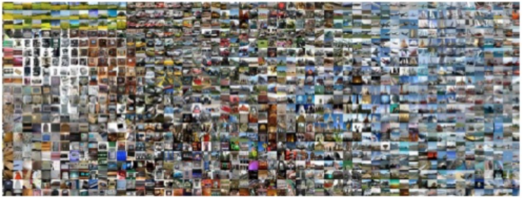
UFR de Mathématiques
Université de Rennes 1

Outline

Outline

- Deep Learning has experienced significant growth over the past decade. It was initially applied to the classification of very large image datasets.
- In 2012, AlexNet (University of Toronto) won the ImageNet competition (1.2 million images, 1000 labels) with a convolutional neural network-based algorithm. The classification error of the winners was around 15
- In 2013, Clarify ConvNet achieved an error rate of 11

IMAGENET



Source: t-SNE visualization of CNN codes. Andrej Karpathy

- Since then, Deep Learning (or deep neural networks) has been applied to a wide range of problems.

- In a conventional Machine Learning procedure, when dealing with high-dimensional data :
  1. We extract features from the data (e.g., through Principal Component Analysis or traditional image analysis techniques). For images, we extract information about texture, colors, brightness, etc.
  2. We fit a model that takes these new variables as input.

- In deep learning, the feature extraction phase is included within the model, typically a neural network. However, the network is then (very) deep and contains a (very) large number of parameters. This requires :
  - A very large dataset to ensure that the number of observations remains significantly larger than the number of parameters.
  - A significant amount of computation time for training.

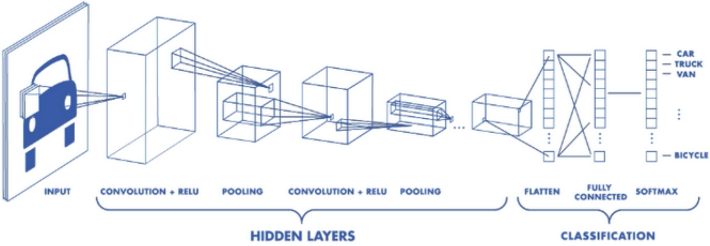
- Below, we will explore two types of deep networks.

This translation provides the text in English while preserving the LaTeX structure.

Outline

Convolutional Neural Networks (CNN)

- One of the primary models used to extract essential features from structured input data (such as curves, images, or videos) is the Convolutional Neural Network (CNN).
- A convolutional network automatically constructs new variables to focus on the features that allow for object recognition.
- This preprocessing step is based on a combination of filtering (convolution) and downsampling (pooling) operations.
- The final part of the neural network is usually a multi-layer perceptron.



A convolutional neural network architecture. Source: Mathworks

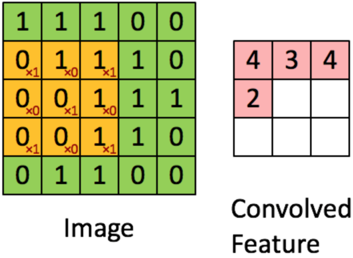
Convolution

Convolutional networks take structured data as input, such as 2D or 3D images or 1D curves. Below, we provide an example using images.

- Each x image of size $r \times c$ is sub-sampled into $a \times b$ patches x_s, and the same filter

$$f_s(x) = \sigma(\omega x_s + \omega_0)$$

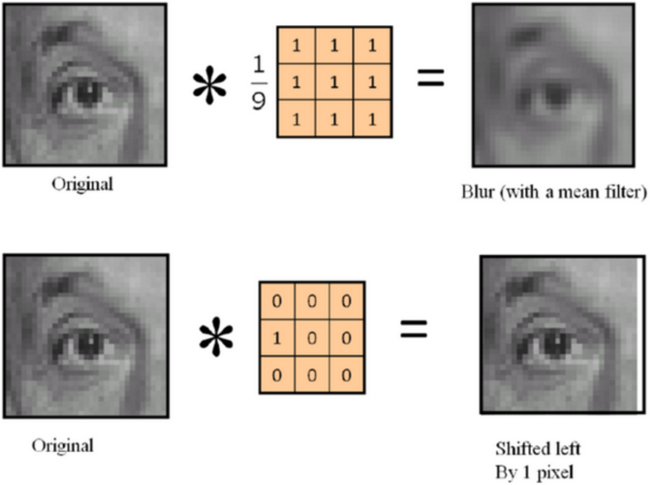
is applied to each patch. The resulting output has size $(r - a + 1) \times (c - b + 1)$.



Link

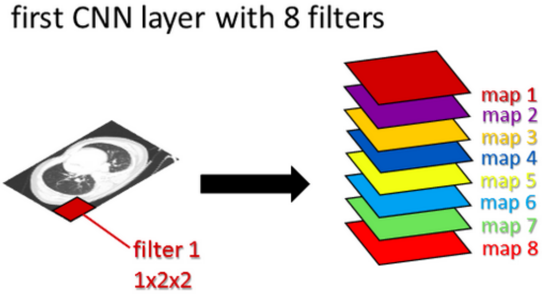
- ω and ω_0 are the weights to be estimated.
- Different values of ω lead to different filters.
- The most common activation function for σ is the ReLu function.

Examples of 3x3 filters



https://towardsdatascience.com/

In practice, at each layer of the CNN, multiple filters of the same size are applied in parallel. The weights of each of the filters are different. This way, different features of the image are explored.



Recap (convoluting)

To summarize, the Conv Layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters :
  - Number of filters N,
  - their spatial extent F (default 3 or 5),
  - the stride S, (default $S = 1$)
  - the amount of zero padding P (default chosen to keep the dimension of the input).
- Produces a volume of size $W_2 \times H_2 \times D_2$ where :

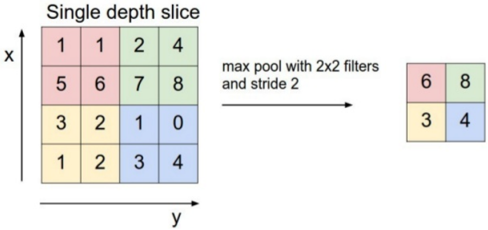
$$W_2 = (W_1 - F + 2P)/S + 1, \quad H_2 = (H_1 - F + 2P)/S + 1, \quad D_2 = K$$

- With parameter sharing, it introduces $F^2 D_1$ weights per filter, for a total of $F^2 D_1 K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Pooling

After the convolution step (or filtering), there is typically a pooling step that reduces the dimension of each feature map.

- The convolved images are divided into patches (e.g., of size 2 by 2), and each patch is replaced by its average (or maximum).
- Example of *max-pooling*



- This transformation does not require any new parameters.

Recap (pooling)

The pooling layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyperparameters :
  - their spatial extent F,
  - the stride S.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where :

$$W_2 = (W_1 - F)/S + 1, \;\; H_2 = (H_1 - F)/S + 1, \;\; D_2 = D_1$$

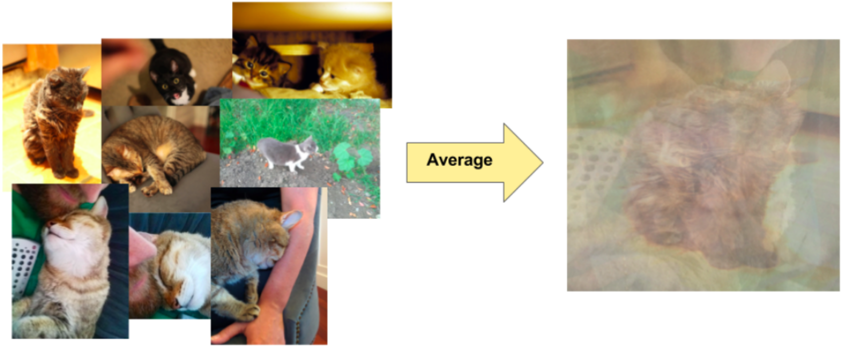
- Introduces zero parameters since it computes a fixed function of the input
- For Pooling layers, it is not common to pad the input using zero-padding.

Outline

Images classification

Several years ago, to classify images, the goal was to identify image features such as background color, shapes, and so on.

However, as seen in the images below, images of the same 'object' can be very diverse. For example, if we calculate their average, no clear trend emerges.



Source https://developers.google.com/machine-learning/practica/image-classification

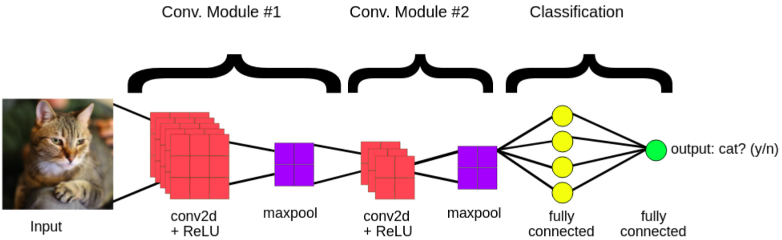
CNN for images classification

- Example

https://drive.google.com/open?id=1A7xgkLqVlWwi40FTqmDNmntsjD2R2SUb

CNN example

Today CNN are used. .



Source `https://developers.google.com/machine-learning/practica/image-classification`

CNN images classification

- Example

https://drive.google.com/open?id=1A7xgkLqVlWwi40FTqmDNmntsjD2R2SUb

Let's build a CNN to classify cats versus dogs.

Exercice 1 : **cliquer ici**

(or copy the link
`https://colab.research.google.com/github/google/eng-edu/blob/main/ml/pc/exercises/image_classification_part1.ipynb`)

Various Stochastic Algorithms

In deep learning, several variants of the stochastic gradient descent (SGD) algorithm have been proposed.

- **Momentum**
  This method involves replacing the gradient with a weighted average of all "historical" gradients. An exponentially weighted forgetting factor is used, resulting in the well-known form of exponential moving average.

$$
\begin{aligned}
m_r &= \alpha m_{t-1} + (1-\alpha)\nabla_{\boldsymbol{\omega}} J(\boldsymbol{\omega}; \mathbf{x}_{i:i+k}, y_{i:i+k}) \\
\boldsymbol{\omega}_{r+1} &= \boldsymbol{\omega}_r - \gamma m_r
\end{aligned}
$$

  Typically, α is chosen to be close to 0.9. This technique is widely used in practice.
- **RMSprop**
  In this variant, adjustments are made to the learning rate.

$$
\begin{aligned}
v_r &= \alpha v_{r-1} + (1-\alpha)\left(\nabla_{\boldsymbol{\omega}} J(\boldsymbol{\omega}; \mathbf{x}_{i:i+k}, y_{i:i+k})\right)^2 \\
\boldsymbol{\omega}_{r+1} &= \boldsymbol{\omega}_r - \frac{\gamma}{\sqrt{v_r + \epsilon}} \nabla_{\boldsymbol{\omega}} J(\boldsymbol{\omega}; \mathbf{x}_{i:i+k}, y_{i:i+k})
\end{aligned}
$$

  The intuition here is that larger steps can be taken when the gradient is small (indicating a flat region of the objective function) and vice versa.
- **Adam**
  The Adam algorithm combines ideas from both Momentum and RMSprop.

Some Vocabulary

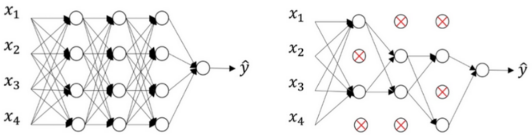
- **Batch** : A small sample of observations (randomly drawn) used to approximate the gradient during a step of the stochastic gradient descent algorithm. Typically, batches have a size of $b = 128, 216, 512$, etc. In practice, the training dataset is randomly shuffled, and the first batch consists of the first b observations, and so on.

- **Epoch** : A cycle of gradient descents during which as many iterations as necessary are performed to evaluate all the batches.

Regularization to Prevent Overfitting

In deep neural networks (including very deep ones), the number of parameters is high, and the risk of overfitting is significant.
Various methods exist for regularization, i.e., reducing the model's complexity during training.

- Penalization
  Similar to Ridge or Lasso regression, you can add an L1 or L2 norm penalty to the loss function.
- Dropout
  At each step of the optimization algorithm, you randomly set a fraction (with probability P) of the weights to zero for that step.



Graph 3. Neural Network with dropout (right) and without (left). Source: Journal of Machine Learning Research 15 (2014)

In linear regression, the average effect of dropout is exactly that of Ridge regression.
Ref : Wager, S. et al. "Dropout Training as Adaptive Regularization." NIPS (2013).

Building a CNN for Cat vs. Dog Image Classification

We will use augmentation and dropout to reduce the risk of overfitting.

Exercise 2 : **Click here**

(or copy the link
`https://colab.research.google.com/github/google/eng-edu/blob/main/ml/pc/exercises/image_classification_part2.ipynb`)

Transfer Learning

By using pre-trained models as a starting point, *Transfer Learning* allows for the rapid development of high-performing models and efficient solving of complex problems in image classification or natural language processing.

In deep learning, the idea is to use models trained on very large datasets and only re-estimate the parameters of the last layers.
The deep layers generally model general image features, and it's the upper layers that specify the class.

Exercise 3 : **Click here**

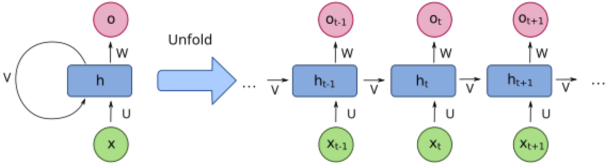
(or copy the link `https://colab.research.google.com/github/google/eng-edu/blob/main/ml/pc/exercises/image_classification_part3.ipynb`)

Outline

Recurrent Neural Networks

Recurrent Neural Network also known as (RNN) that works better than a simple neural network when data is sequential like Time-Series data and text data.

RNN is a type of Neural Network where the output from the previous step is fed as input to the current step.



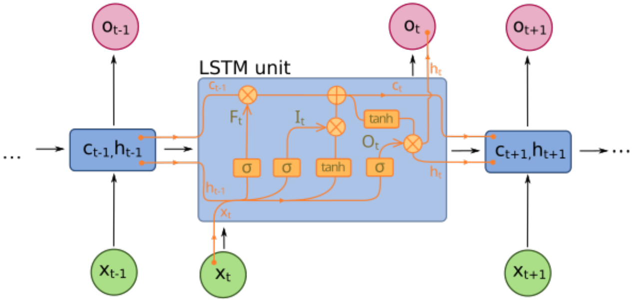
RNNs come in many variants. Abstractly speaking, an RNN is a function f_θ of type $(x_t, h_t) \mapsto (y_t, h_{t+1})$, where x_t : input vector, h_t : hidden vector, y_t : output vector, θ : neural network parameters.

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

σ_h is the tanh function.

Long short-term memory

Long short-term memory (LSTM) is the most widely used RNN architecture. It helps to solve the problem of vanishing gradient [1].



1. The hyperbolic tangent function have gradients in the range [-1,1], and backpropagation computes gradients by the chain rule. This has the effect of multiplying n of these small numbers to compute gradients of the early layers in an n-layer network, meaning that the gradient (error signal) decreases exponentially with n while the early layers train very slowly.

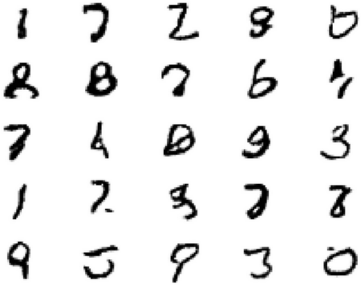
Outline

Generative Adversarial Networks (GAN)

Another important application of deep neural networks is the generation of images (or speech, text, etc.).

A GAN is a deep neural network that generates data, labeled or not, that are indistinguishable from real data samples.

For example, GANs can be used to create images from noise.



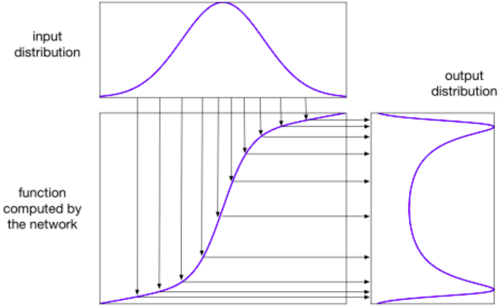
Examples of digits (simplistic GAN trained for 100 epochs) or to modify/improve existing images.

Image generation is an unsupervised problem.

GAN, a Random Generator

Think of GANs as random generators that draw samples from a target distribution (e.g., the probability distribution of real images).

To simplify, we can illustrate the problem in 1D.



In general, the target space is of (very) high dimension, and we don't have explicit access to the target distribution.

Generator Architecture

For the digits example, the generator is constructed using the following architecture :

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 6272)              633472
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 6272)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 14, 14, 128)       262272
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 128)       262272
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 28, 28, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 1)         6273
=================================================================
Total params: 1,164,289
Trainable params: 1,164,289
Non-trainable params: 0
```

Training a GAN

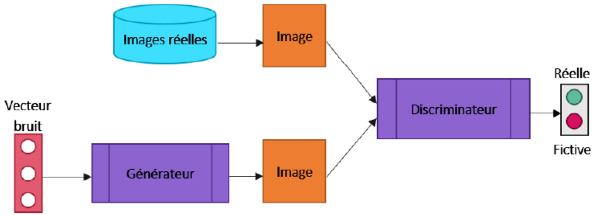
To train the generator, consider the following structure composed of the generator and a discriminator.
The generator creates images from source images or noise.

$$G : z \in \mathbb{R}^{p'} \mapsto G_{\theta_G}(z) \in \mathbb{R}^p$$

The discriminator is a binary classifier that returns 1 if the image is real and 0 otherwise.

$$D : x \in \mathbb{R}^p \mapsto D_{\theta_D}(x) \in \{0, 1\}$$

It indicates to the generator whether the fake images have the same characteristics as real images.



Loss Function

In a GAN, the competition between the generator G and the discriminator D is expressed as a contradictory loss function that measures the divergence between the distributions of the generator's outputs and real images :

$$L(\theta_G, \theta_D) = E_X\left[-\log D(X)\right] + E_Z\left[-log(1 - D(G(Z)))\right]$$

where E_Z is the expectation over Z.

If the discriminator has low entropy, it means it can easily distinguish between real and fake ; if it has high entropy, it means it cannot. Therefore, the simplest criterion for the generator is to maximize the cross-entropy of the discriminator :

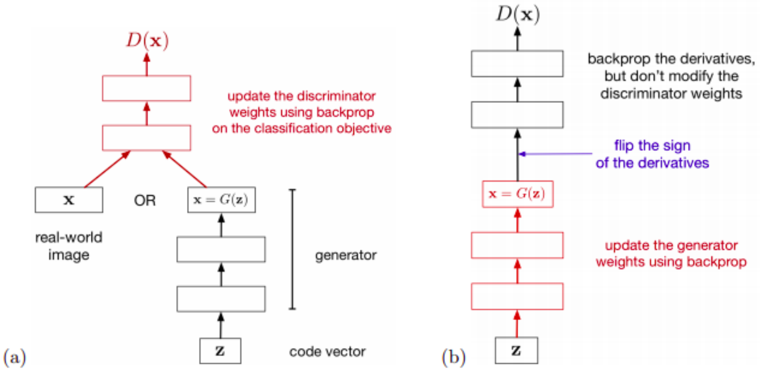
$$L(\theta_G, \theta_D) \sim E_X\left[log(1 - D(G(X)))\right]$$

The weight vector (θ_G, θ_D) is a solution to a minimax problem.

$$(\theta_G^*, \theta_D^*) = \arg \max_{\theta_G} \min_{\theta_D} L(\theta_G, \theta_D)$$

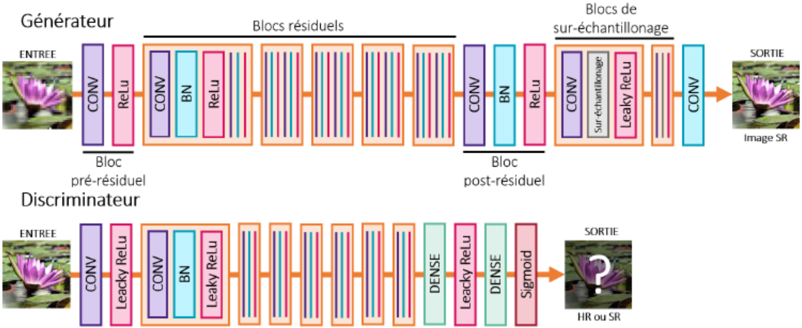
Simultaneous Training

The generator and discriminator are trained simultaneously by repeating two steps until convergence.
In the first step, we train the discriminator with the generator fixed, and in the second step, we update the generator's weights with the discriminator fixed.



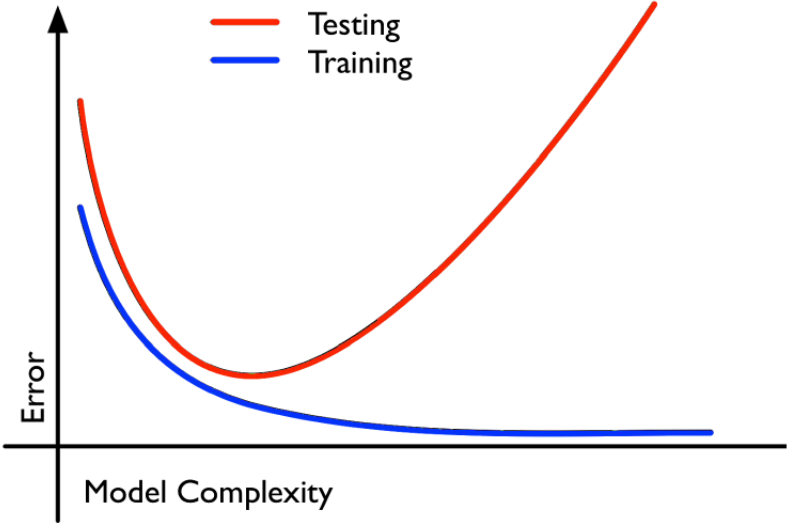
Example

Architecture example used to increase the resolution of images.



Outline

What Does Theory Say ?



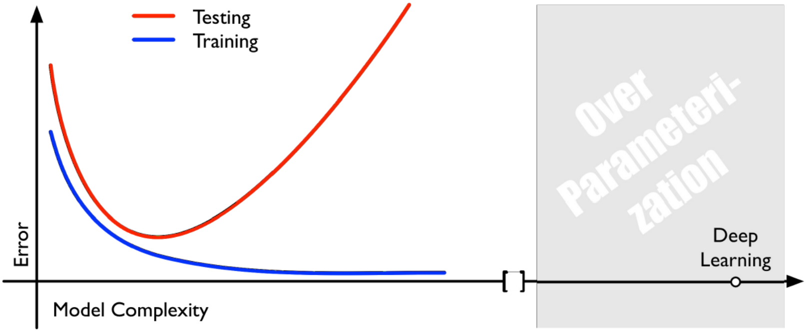
Number of Parameters in Classic Deep Networks

CIFAR-10	**# train: 50,000**
Inception	1,649,402
Alexnet	1,387,786
MLP 1x512	1,209,866
ImageNet	**# train: 1,200,000**
Inception V4	42,681,353
Alexnet	61,100,840
Resnet-{18;152}	11,689,512; 60,192,808
VGG-{11;19}	132,863,336; 143,667,240

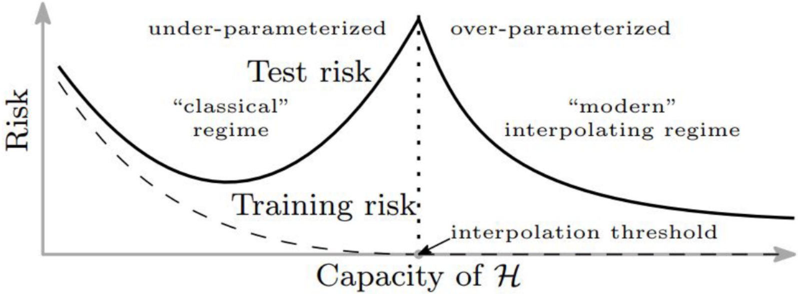
What Happens if Labels Are Assigned Randomly ?

Model	Random Label	train accuracy	test accuracy
Inception-small	No	100%	85.75%
	Yes	100%	9.78%
Alexnet	No	100%	76.07%
	Yes	99.82%	9.86%
MLP 3x512	No	100%	52.39%
	Yes	100%	10.48%
MLP 1x512	No	100%	50.51%
	Yes	99.34%	10.61%
Inception V3	No	100%	80.38% (84.49%)
	Yes	99.14%	0.56%
Alexnet (2012)	No	-	83.60%

What Does Theory Say ?



Beyond the Interpolation Threshold...



Outline

Deep Learning in Python

In Python, there are essentially 2 toolkits for deep learning : Keras and PyTorch.

For the labs, we will use Keras. Make sure to install it in advance if you are using a local Python distribution ! An alternative is to use Google Colab or Kaggle.

Keras : Installing Keras is not always simple. The best approach is to consult Keras' documentation and follow the installation instructions : (`https://keras.io/#installation`).

Outline

Conclusion

- Neural networks are regressors or classifiers with a particular structure.
- Their structure allows, in theory, to approximate all sorts of functions.
- This particular structure also enables the use of efficient numerical techniques to compute gradients.

- Deep neural networks are networks in which a very large number of perceptrons are combined (in series and in parallel).
- They are used in a wide range of applications.
- The number of parameters in these networks is usually much larger than the number of observations ; yet, the performance of these models is often excellent.
- This takes us beyond the classical framework of statistical estimation, and recent (very) recent results show that these deep networks belong to a world that goes beyond interpolation. To be continued...