Introduction to Deep Learning

A practical approach

Mikaël Jacquemont

Laboratoire d'Annecy de Physique des Particules - IN2P3
Univ. Savoie Mont Blanc, CNRS

ESCAPE Summer School 2021

Objectives and outline

Goals of the lecture:

- Give a (narrow) overview of deep learning and its process
- Provide some keys to step in
- Explain into more details the fundamental components
- Experiment with widespread deep learning tools

At the end of the lecture, you should be able to:

- Design, implement and train a simple neural network
- Load a pretrained very deep network and fine tune it

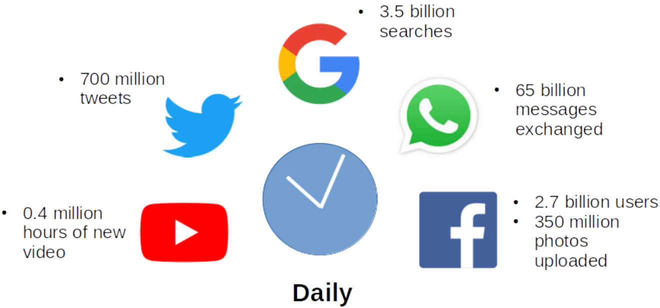
Objectives and outline

Outline

- Some elements of context
- Part I, Deep learning fundamentals
  - Basic building blocks
  - Training process
  - Deep learning stack (hardware and software)
- Part II, Going (really) deep
  - OVerview of famous very deep architectures
  - Transfer learning: being lazy is good for the performance (and the planet)
  - Introduction to explainability
- Wrap up and (some) further reflections

Context
A world of (big) data

- Internet generates 2,500 Peta Bytes of data daily !



[Gro20], [Pet21], [Asl21]

Context
A world of (big) data

But also (scientific) specific fields:

- Medical imaging (43 million MRI scans in Europe in 2017 [Eur20])
- Remote sensing (Sentinel satellites provide hundreds of GB of data / year)
- Astronomy
  - Gamma astronomy (CTA will generate 210 PB of data / year [co])
  - Vera C. Rubin Observatory will produce 10 TB of data / night for 10 years [Obs]
- and many other domains ...

Context
A world of (big) data

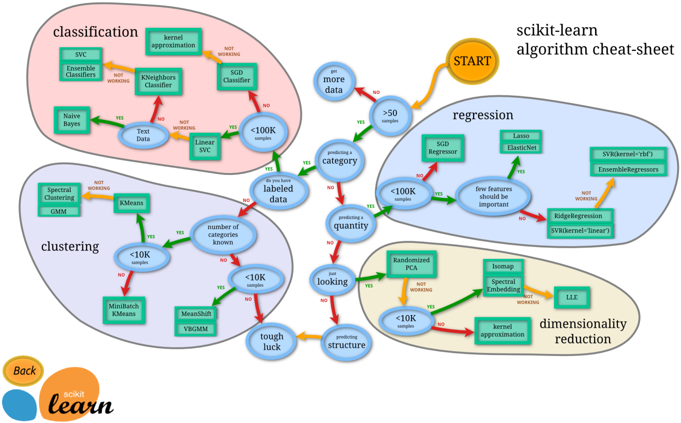
From these data, we want:

- To sell advertises (18 billion \$ revenue for Facebook between April and June 2020 [Fac21])
- Our cars to self-drive
- To help doctors identify disease
- To improve agriculture
- To better understand the Universe
- ...

We need machine learning to process these big data

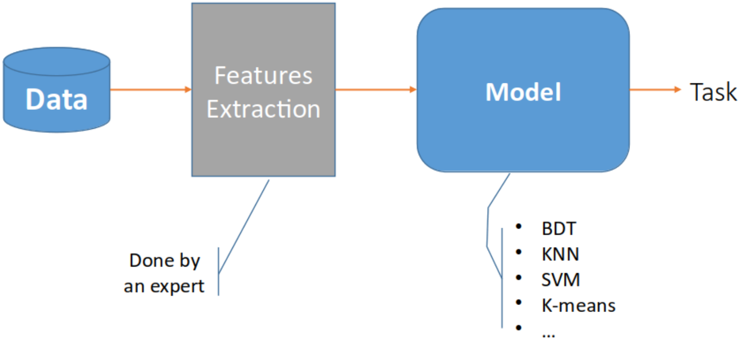
Context
Machine learning, a quick reminder

Machine learning = algorithms learn to solve tasks



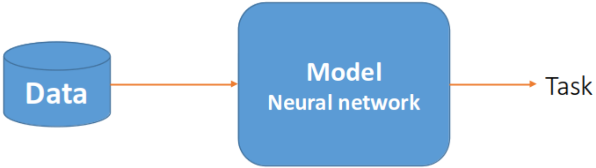
Supervised vs Unsupervised

Machine learning principle



Context
Machine learning, a quick reminder

~~Machine~~ learning principle
Deep



- The features are learned by the model
- Search for regularity with an over-parametrized model within large sets of data

Part I, Deep Learning fundamentals

Disclaimer

In this lecture, we mainly focus on:

- Computer vision tasks and models
- Supervised learning (classification and regression)

But the techniques presented are fully applicable to other fields / learning methods.

In the following, important vocabulary will be highlighted in red.

Part I, Deep Learning fundamentals

Outline

- A bit of history
- Building deep neural networks
- Training deep neural networks
- Deep learning libraries
- First hands on session

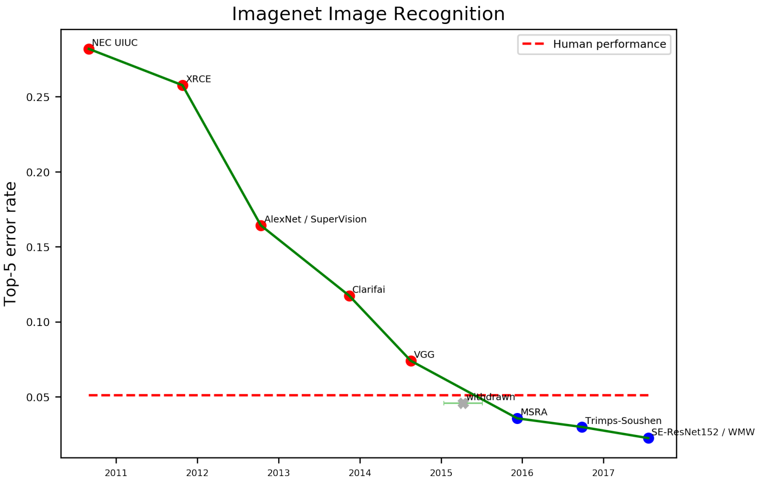
Part I, Deep Learning fundamentals

Spoiler alert ! This is **not** the current state of AI / DL

Chappie, 2015

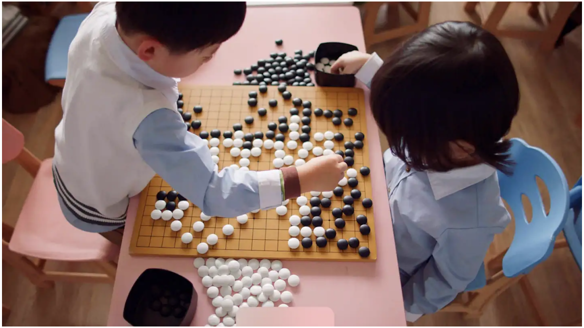
- DL is far from human-level AI
- It lacks:
  - Generalization
  - Causality
  - Logical reasonning

- DL outperforms humans on **specific and narrow** tasks



Part I, Deep Learning fundamentals

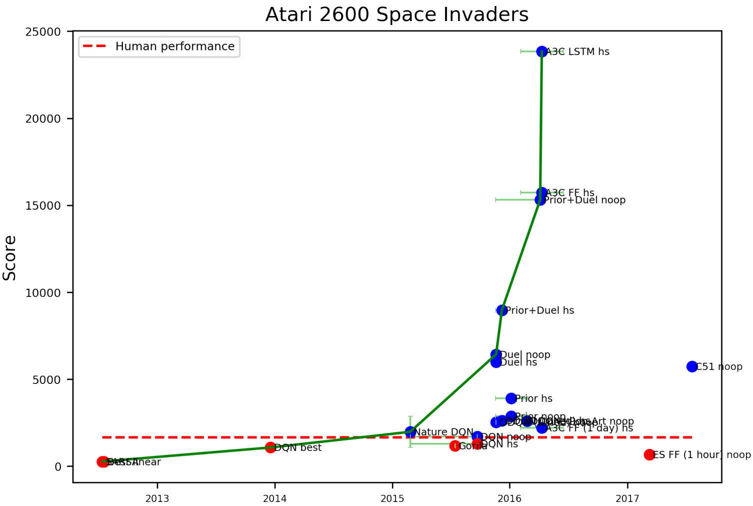
- DL outperforms humans on **specific and narrow** tasks



AlphaGo

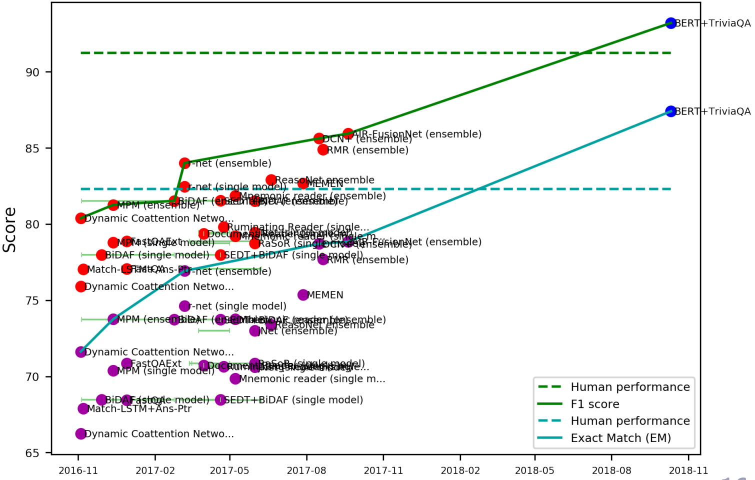
Part I, Deep Learning fundamentals

- DL outperforms humans on **specific and narrow** tasks



- DL outperforms humans on **specific and narrow** tasks

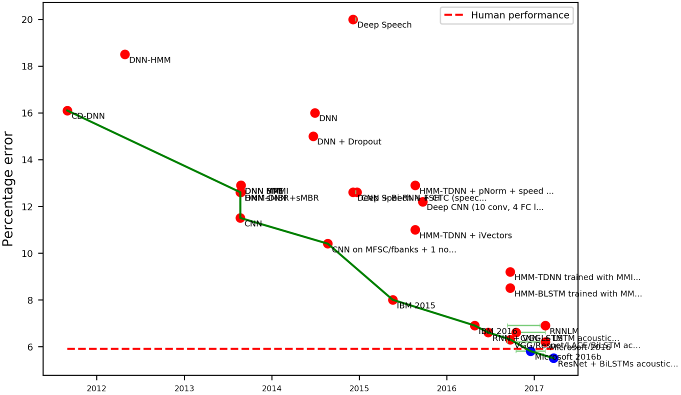
Stanford Question Answering Dataset (SQuAD)



Part I, Deep Learning fundamentals

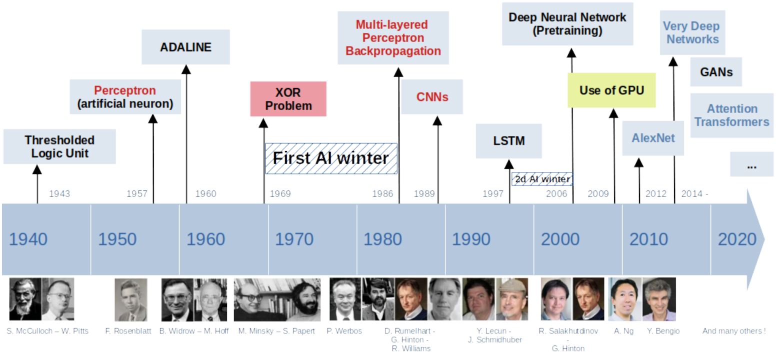
- DL outperforms humans on **specific and narrow** tasks

Word error rate on Switchboard trained against the Hub5'00 dataset

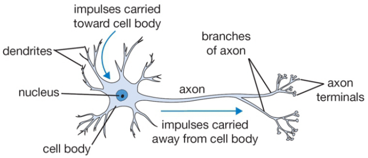


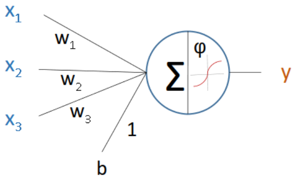
Source

Part I, Deep Learning fundamentals
From the artificial neuron to deep learning



The artificial neuron, or perceptron, is the fundamental component of neural networks

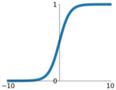




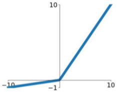
- Inspired from the biological neuron
- Simple mathematical operation followed by a non-linear activation function: $y = \varphi(\sum_i w_i \cdot x_i + b)$
- Learning ability comes from the trainable parameters w_i and b, also named weights

There exists a large variety of activation functions. Some popular ones:

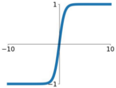
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



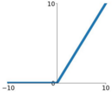
Leaky ReLU
$\max(0.1x, x)$



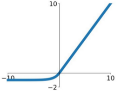
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

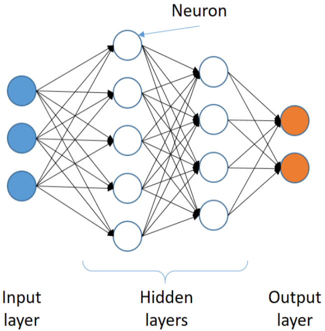


ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



- Sigmoid and tanh are *historical* activation functions but they saturate → vanishing gradient
- ReLU and its derivatives solve the vanishing gradient issue

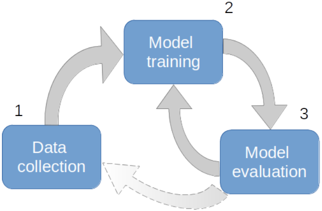
Neural networks are orgnized as layers of neurons:



- This kind of neural network is a multilayer perceptron composed of fully connected layers
- Deep learning $\rightarrow$ 3 hidden layers or more

The training process

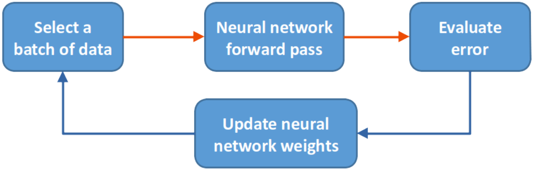
1. Collect and prepare the training data
2. Train the model with all the data $\rightarrow$ 1 epoch
3. Evaluate the model performance
4. If necessary, repeat steps 2 and 3



- Neural networks generally require from **tens to hundreds epochs**

Going into the details of step 2 (model training)

- Data generally don't fit in memory → **batch** of data
- Training the model with a batch is an **iteration**
- The iteration process:
  1. Neural network forward pass = prediction on the batch
  2. Evaluate the prediction error
  3. Update neural networks weights to minimize the prediction error (backward pass)
- We repeat this process until all the data are consumed $\rightarrow N_{iteration} = \frac{N_{data}}{N_{batch}} \approx hundreds \rightarrow thousands$



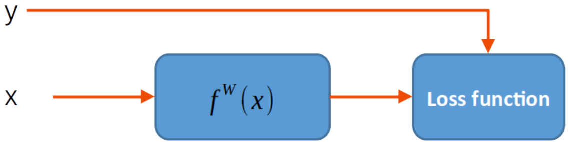
Going into more details

- We evaluate the error with a loss function
- For supervised learning
  - Classification: Cross-Entropy (Softmax + NLL)

    $L = \frac{1}{n}\sum_n \sum_i -y_{true,i} \cdot \log(p(c_i = 1|x))$ with

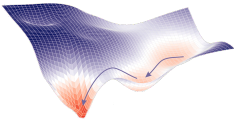
    $p(c_i = 1|x) = \frac{\exp^{y_{pred,i}}}{\sum_j \exp^{y_{pred,j}}}$ and $y_{true} = [\overset{1}{0}0\ldots0\overset{i}{1}0\ldots\overset{c}{0}]$
  - Regression: MSE

    $L = \frac{1}{n}\sum_n (y_{pred} - y_{true})^2$



- Key algorithm to update the weights and minimize the loss: stochastic gradient descent (SGD) $\rightarrow$ optimizer
- $W_{t+1} = W_t - \eta \cdot \nabla_{W_t} L$ with η the learning rate

- Challenges
  - Finding a "good" local minimum
  - Choosing the lr is difficult
  - The initial lr might not be optimal during the whole training process
  - The same lr might not be optimal for all the parameters
  - Avoiding saddle point is crucial

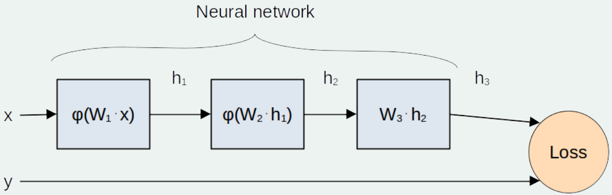


- Several evolutions: momentum, adative methods including Adam ...
- See [Rud16] for details

- But how can we apply SGD to **millions** of parameters ?
- Solving analytically $\nabla_{W_t} L$ is untractable
- Solution: the backpropagation algorithm !

Example

Consider the following neural network:



with $\varphi(a) = max(0, a)$ and a Cross-entropy loss

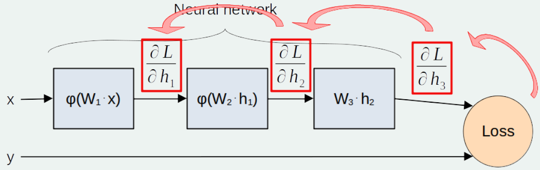
For gradient descent, we need to compute $\frac{\partial L}{\partial W_1}$, $\frac{\partial L}{\partial W_2}$ and $\frac{\partial L}{\partial W_3}$

The gradient of the loss w.r.t. the last features h_3 is:

- $\frac{\partial L}{\partial h_3} = p(c|x) - y_{true}$

Then, with the chain rule, we can compute:

- $\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W_3} = (p(c|x) - y_{true}) \cdot h_2^T$
- $\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W_2} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W_2}$
- $\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial W_1} = \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_1}$



Vocabulary break

- The parameters are the trainable weights of the network
- The hyperparameters are defined by the expert to set up the experiment, including:
  - The network definition (number, size and type of each layer)
  - The choice of the optimizer (SGD, Adam . . .)
  - The learning rate and its evolution during the training
  - The loss function
  - . . .

Part I, Deep Learning fundamentals
Ressources needed

We are able to design and train neural networks to do fancy stuff, but:

- Neural networks can have millions of parameters (the heaviest architecture has currently 1.6 trillion !!)
- The training process implies thousands of iterations and tens to hundreds epochs
- The hyperparameter space is huge !

→ We need efficient computing ressources

Part I, Deep Learning fundamentals
Ressources needed

Fortunately, most DL computations can be carried out in parallel
→ we can benefit from GPUs (A. Ng et al., 2009)

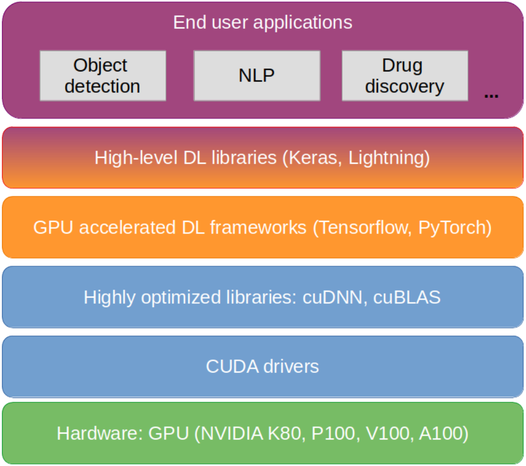
- GPUs are part of the success of neural networks
- The training time is divided by up to 70 compared to CPU
- Modern GPUs are composed of thousands of processing units, some specialized in DL operations (e.g., Tensorcores)
- NVIDIA CUDA (+ cuDNN) drivers contain very optimized algorithms for DL fundamental operations (matrix multipication, convolution. . .)

Part I, Deep Learning fundamentals
Ressources needed

We also need highly efficient software libraries:

- To efficiently exploit hardware capabilities
- To offer high level abstraction to deep learning components
- To take care of the computation graph and gradient operations
- To manage engineering code and focus on research (hyperparameter definition)

General deep learning stack



Part I, Deep Learning fundamentals
Deep learning libraries

There exists many DL frameworks

- The elder ones: Caffe, Theano, Torch
- The outsiders: Chainer, Deeplearning4j, MXNet, Caffe2
- The leaders:
  - Tensorflow (static graph)
  - **PyTorch** (dynamic graph)

They offer interfaces in several languages: **Python**, C++, Java, Javascript

PyTorch

TorchVision

TorchAudio

TorchText

- Popular datasets
- Model architectures (pre-trained)
- Common data transformations

Ecosystem
(3^{rd} party libraries)

- Numpy + CUDA
- Autograd
- DL operations

PyTorch hub
(3^{rd} party models)

TorchServe

PyTorch quickstart

- Basic data structure: PyTorch tensor $\rightarrow$ Numpy array $+$ CUDA $+$ autograd

```
import torch
import numpy as np

np_array = np.array([0, 1, 2])
torch_tensor = torch.from_numpy(np_array)
torch_tensor
> tensor([0, 1, 2])
torch_tensor.requires_grad
> False
```

- By default, tensors **do not** retain gradient information

Working with data

- Standard vision datasets can be loaded from TorchVision

```
from torchvision.datasets import CIFAR10
from torchvision.transforms import ToTensor

dataset = CIFAR10(root='./data', train=True,
                  download=True,
                  transform=ToTensor())
```

- The output of torchvision datasets are PILImage images
- We need to transform them to Tensors
- We can implement custom datasets by inheriting Dataset class

Working with data

- DataLoader takes care of sampling data batches during training

```
from torch.utils.data import DataLoader
training_data = DataLoader(dataset,
                           batch_size=32,
                           num_workers=4,
                           shuffle=True)
```

Creating neural networks

- We find neural network layers in torch.nn

```
from torch import nn
nn.Linear(in_features=512, out_features=512)
nn.ReLU()
nn.Conv2d(in_channels=3, out_channels=16,
          kernel_size=3, padding=1)
```

- We can build simple models as Sequential

```
model = nn.Sequential(
    nn.Linear(in_features=32, out_features=16),
    nn.LeakyReLU(),
    nn.Linear(in_features=16, out_features=10)
)
```

- For more complex models

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 10),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

Part I, Deep Learning fundamentals
First hands on session

Time to work !

3 tutorials to smoothly enter the deep learning world:

- T1: Fitting a noisy sinusoid with a MLP
- T2: Training a MLP on a binary classification case
- T3: Training a MLP on a multi-class classification case

Part II, Going (really) deep

Part II outline

- Convolutional neural network: overcoming the limitations of MLP
- Famous very deep networks and modern architectures
- Transfer learning: recycling knowledge
- Introduction to explainability
- High-level deep learning library and logging tool

Part II, Going (really) deep
Convolutional neural networks

The MLP is a universal function approximator [Cyb89], but:

- The number of parameters can rapidly grow with the size of the input and the number of layers
  $\rightarrow N_i = (n_{i-1} + 1) \times n_i$

Example

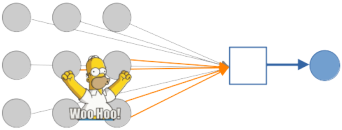
CIFAR10 dataset: 

- Small RGB images of 32x32 pixels
- Fully connected $\rightarrow$ each pixel is a feature
- **Each neuron** of the first hidden layer would have 32x32x3 + 1 = 3073 parameters !

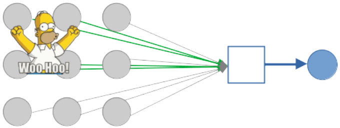
Part II, Going (really) deep
Convolutional neural networks

- MLP is not translation invariant → severe drawback for computer vision

The orange weights are modified to better recognize Homer



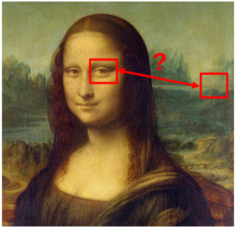
The green weights are modified to better recognize Homer



→ Learning redundant features + not robust

In images,

- Nearby pixels are more strongly related than distant ones
- Objects are built up out of small parts



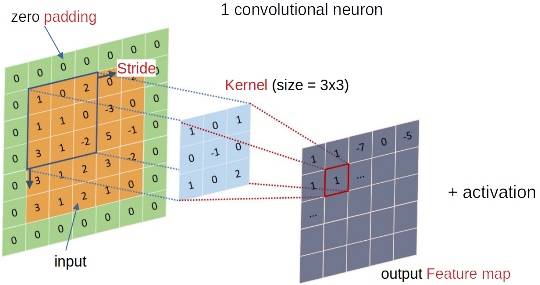
Translates to other fields, e.g., text analysis:

Towns and cities across Scotland would be devastated if the country's coastline was hit by a tsunami of the kind that happened 8,200 years ago, according to an academics' study. While about 370 miles of Scotland's northern and eastern coastline were affected when the Storegga tsunami struck, the study suggests a modern-day disaster of the same magnitude would have worse consequences. (source: The Guardian)

Part II, Going (really) deep
Convolutional neural networks

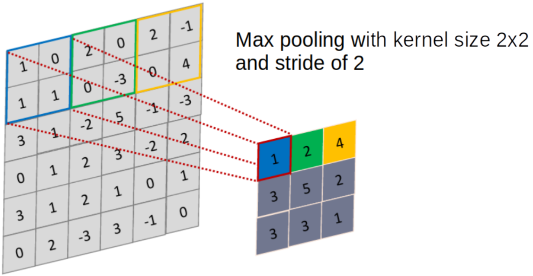
Solution: the convolution (derives from the visual cortex)

- The convolutional neuron operates on a local neighborhood → shared weights = convolution kernel
- The kernel slides over the input with a chosen stride → 1 position = 1 pixel of the output feature map
- 1 convolutional neuron → 1 output feature map
- Convolution layer = stack of n neurons → n feature maps



The pooling operation

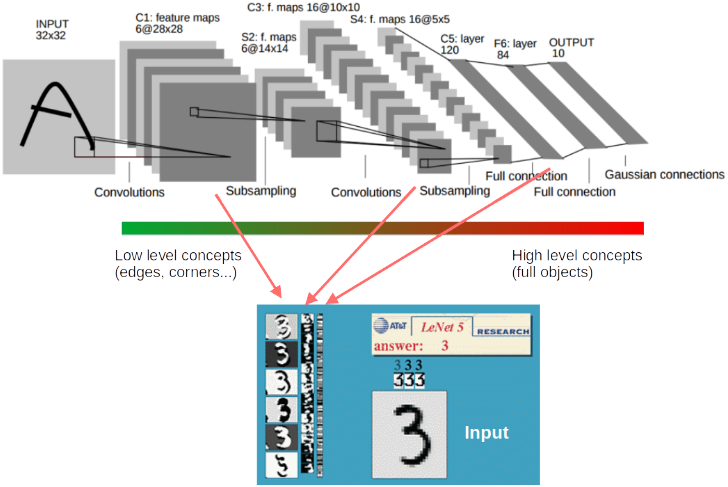
- Also operates on a local neighborhood
- A form of non-linear down-sampling (with stride > 1)
- Participates to (local) spatial invariance
- Standard pooling functions: max, average or convolution



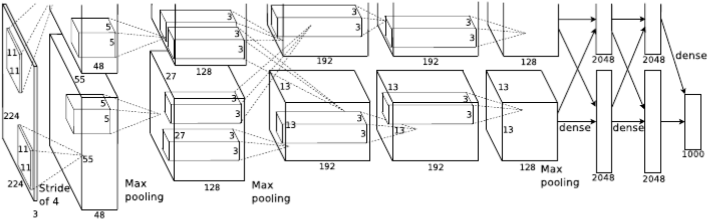
Building a convolutional neural network (CNN)

- CNN = encoder + decoder
- Encoder
  - Learns a (compressed) latent representation of the data
  - A lasagna of convolution layers (including activations) and pooling layers
- Decoder
  - Learns to solve the task from the latent representation
  - One or more fully connected layers
- Advantages of CNN over MLP
  - The total number of parameters is dramatically reduced
  - Translation invariance
  - Pixel position and neighborhood have semantic meanings

Example: LeNet-5 for character recognition (1998)



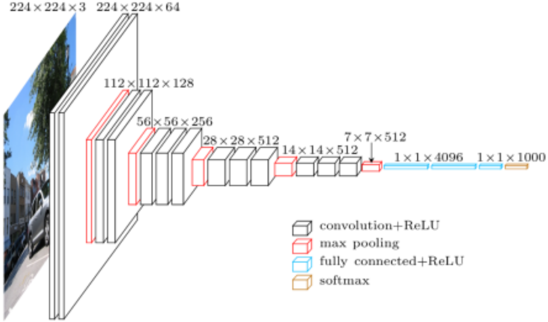
AlexNet, the pioneer: beating standard CV methods in the ImageNet Large Scale Visual Recognition Challenge 2012 [KSH12]



- 5 convolution layers
- ReLU activations
- Dropout

- Multi-GPUs training
- Top-5 error: 15.3% vs 26.1%

VGG, starting to go deep [SZ15]

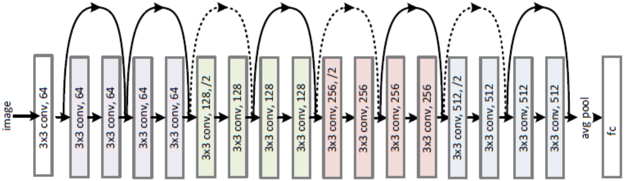


VGG-16

- Up to 19 layers
  → (144 M parameters)
- 3×3 convolution kernels

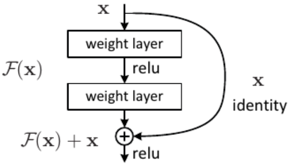
- Group of convolutions followed by max-pooling
- Top-5 error on ILSVRC 2012: 6.8%

ResNet, going really deep with residual connections [HZRS16b, HZRS16a]



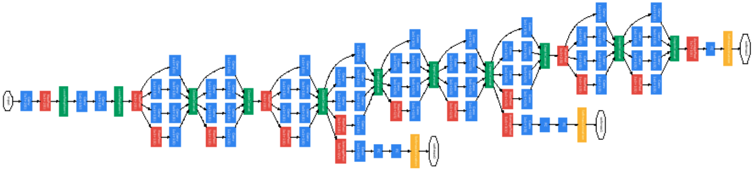
ResNet-18

- Up to 1202 layers !
- Too many layers cause gradient vanishing issue
  → Residual connections
- Top-5 error on ILSVRC 2012: 4.8% (ResNet-200)



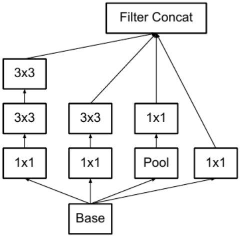
Part II, Going (really) deep
Famous deep learning architectures

Inception, going deep and wide [SLJ+15, SVI+16]



Inception-V1

- Multiple-size filters on the same level
- 42 layers (V3) but faster than VGG
- Top-5 error on ILSVRC 2012: 4.2% (Inception-V3)

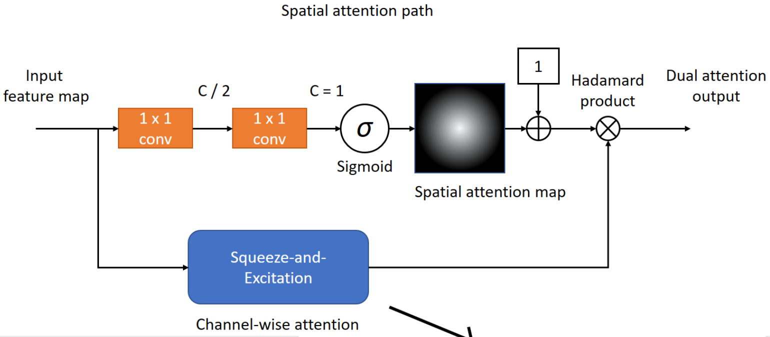


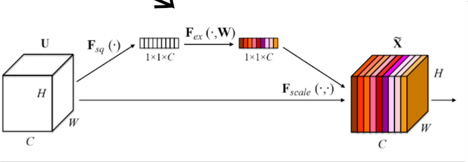
Inception-V3 module

Attention is all you need !

- Helps the model focus on relevant features
- Originates in NLP: main component of Transformer architecture [VSP+17, DCLT19] $\rightarrow$ SOTA in neural machine translation and image captioning
- Increasing use in computer vision architectures
- Standalone or in combination with convolution block (e.g., ResNet-SE [HSS18])
- Attention types in CV
  - pixel-wise attention, also named spatial attention
  - channel-wise attention
  - combination of both

Dual Attention [SDZW20] example





Part II, Going (really) deep
Transfer learning

Training modern DL architectures from scratch for a specific application is costly. It requires:

- Huge amount of data
- Powerful GPUs
- Time to tweak the hyperparameters

Training the Transformer [VSP+17] for NLP, including architecture search, has emitted 284 tonnes of CO_2 ! [SGM19]

Then, let's try to recycle with **transfer learning** !

- Many SOTA algorithms trained on standard data sets are available !
- We can continue the training on our (similar) specific task (and our little data)
  $\rightarrow$ Leveraging previously learned features

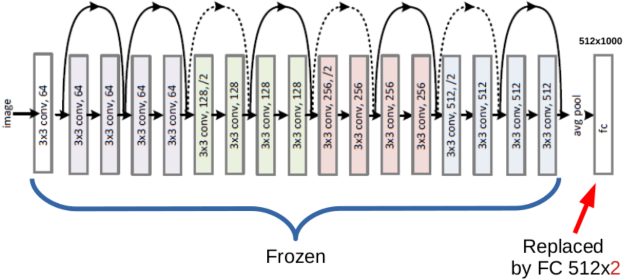
Part II, Going (really) deep
Transfer learning

Transfer learning steps

1. Select a model trained on a task similar to your problem
2. Load layers from trained model
3. Freeze most of them (generally the first ones) to keep knowledge
4. Add / change layers to match your task
5. Train the new and unfrozen layers

Transfer learning example

- Task: classify between screws and nuts ($\sim$ 500 images)
- Model: ResNet-18 trained on ImageNet (1000 classes)
- Transfer learning
  1. Freeze all the convolutional layers
  2. Replace the last FC layer to match our 2 classes task
  3. Train this last layer on our little data set of screws and nuts



Deep learning models are used in several fields with high impact on our lives:

- Healthcare
- Insurance
- Autonomous vehicles
- Forensics
- Candidate selection for jobs
- ...

We need to understand the model's decision !

- Interpretability: how a drift in the data affects the decision
- Explainability: better understand the internal mechanics that leads to the decision

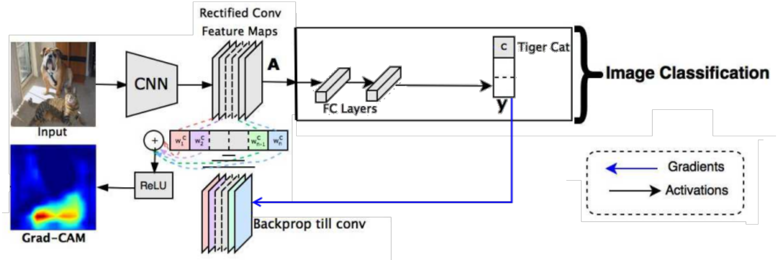
Part II, Going (really) deep
Introduction to explainability

Explainable deep learning is a highly active research domain [XRvGD20].

3 types of method:

- Model distillation: develop a separate explainable model (such as random forests) that mimic the behavior of the neural network
- Intrinsic methods: make use specific layers such as attention layers
- Visualization methods: highlight the most important pixels for model's decision
  - Perturbation-based: comparison of network decision between an input and an altered copy of the input
  - Gradient-based (or backpropagation-based)

Gradient-based visual explanation method example: Grad-CAM



1. Compute the gradient of the output of interest (e.g., 'Tiger Cat') w.r.t. the convolutional feature maps (A)
2. Average each gradient map to obtain a coefficient
3. Sum the feature maps A weighted by their respective coefficient
4. Apply ReLU activation to only keep positive values

Deep learning is a very empirical and iterative process
$\rightarrow$ We need to train a lot of different models !
To ensure traceability and reproducibility, several tools exist. In the second hands-on we'll use:

- PyTorch Lightning: takes of the engineering operations to let us focus on science (architectures and hyperparameters)
- Tensorboard (a dashboard to watch them all): allows for an easy comparison of performance metrics between models

Part II, Going (really) deep
Second hands on session

Time to work !

2 more tutorials to go deeper:

- T4: CIFAR10 classification with a sipmle CNN
- T5: Transfer learning for Cats and Dogs classification with a ResNet

Wrap up

In this lecture, we have learned:

- what is an artificial neuron
- how to arrange neurons in layers
- how to combine layers to build neural networks
- how to train neural networks in a supervised manner
- what is a convolution and why it is crucial for computer vision tasks
- how to apply transfer learning to benefit from already trained models

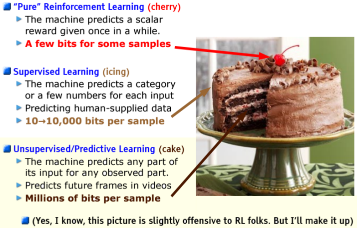
We have also discovered famous architectures and experimented with tools to ease the DL process.

Beyond this introduction, there is way more to discover !

Wrap up

Widening the sphere of possibilities

- From CV to many other application fields: NLP, Autonomous agents, Speech processing, Smart cities, Particle physics and astrophysics . . .
- From classification/regression to many other learning techniques: GANs, RL, Unsupervised and semi-supervised learning



Y. Lecun

Wrap up

Widening the sphere of possibilities

- Improving training and reducing overfitting (batchnorm, dropout, regularization...)
- Changing the AI paradigm into a **data-centric** benchmark

> AI Systems = Code (model/algorithm) + Data.
>
> **Most academic** benchmarks/competitions hold the **Data fixed**, and let teams work on the Code. Thinking of organizing something where **we hold the Code fixed, and ask teams to work on the Data**.
>
> A. Ng

Wrap up

Widening the sphere of possibilities

> To extend deep learning to reach **human-level AI**, we are missing out-of-distribution generalization, higher-level cognition (system 1 to system 2) and agent perspective.
>
> Y. Bengio

System 1 cognition

- Intuitive, fast, **Unconscious**
- Current DL

→

System 2 cognition

- Logical, slow, **Conscious**, reasoning
- Future DL

Y. Bengio presentation at NeurIPS 2019

Wrap up

Usefull links

- The future of deep learning: Deep Learning for AI
- To better understand backpropagation: https://colah.github.io/posts/2015-08-Backprop/ and https://ranzato.github.io/publications/ranzato_deeplearn17_lec1_vision.pdf (from slide 53)
- A recipe for training neural networks: http://karpathy.github.io/2019/04/25/recipe/
- Play with simple neural networks: https://playground.tensorflow.org
- https://pytorch.org/
- https://www.pytorchlightning.ai/

References I

Salman Aslam, Facebook by the numbers: Stats, demographics & fun facts, https://www.omnicoreagency.com/facebook-statistics/, 2021, Accessed: 2021-05-28.

cta observatory, Data acquisition, processing, archiving and access, https://www.cta-observatory.org/project/technology/computing/, Accessed: 2021-05-28.

George Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems **2** (1989), no. 4, 303–314.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

Eurostat, Healthcare resource statistics - technical resources and medical technology, https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Healthcare_resource_statistics_-_technical_resources_and_medical_technology, 2020, Accessed: 2021-05-28.

Facebook, Facebook reports second quarter 2020 results, https://investor.fb.com/investor-news/press-release-details/2020/Facebook-Reports-Second-Quarter-2020-Results/default.aspx, 2021, Accessed: 2021-05-28.

Saas Scout Research Group, Big data statistics, growth & facts 2020, https://saasscout.com/statistics/big-data-statistics/, 2020, Accessed: 2021-05-28.

Jie Hu, Li Shen, and Gang Sun, Squeeze-and-excitation networks, Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

References II

- Kaiming He, Jiangyu Zhang, Shaoqing Ren, and Jian Sun, Identity mappings in deep residual networks, European conference on computer vision, Springer, 2016, pp. 630–645.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep residual learning for image recognition, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems, 2012, pp. 1097–1105.
- Vera C. Rubin Observatory, Opening a window of discovery on the dynamic universe, https://www.lsst.org/, Accessed: 2021-05-28.
- Christo Petrov, 25+ impressive big data statistics for 2021, https://techjury.net/blog/big-data-statistics/, 2021, Accessed: 2021-05-28.
- Sebastian Ruder, An overview of gradient descent optimization algorithms, 1–14.
- Jesse Sun, Fatemeh Darbeha, Mark Zaidi, and Bo Wang, Saunet: Shape attentive u-net for interpretable medical image segmentation, arXiv preprint arXiv:2001.07645 (2020).
- Emma Strubell, Ananya Ganesh, and Andrew McCallum, Energy and policy considerations for deep learning in nlp, arXiv preprint arXiv:1906.02243 (2019).

References III

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Chapel Hill, and Ann Arbor, Going Deeper with Convolutions, Proceedings of the IEEE conference on computer vision and pattern recognition (2015), 1–9.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, Rethinking the inception architecture for computer vision, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

Karen Simonyan and Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (Yoshua Bengio and Yann LeCun, eds.), 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, Attention is all you need, Advances in neural information processing systems, 2017, pp. 5998–6008.

Ning Xie, Gabrielle Ras, Marcel van Gerven, and Derek Doran, Explainable deep learning: A field guide for the uninitiated, arXiv preprint arXiv:2004.14545 (2020).

IDEFICS

Intro to DL

Mikaël J.

This event is organized in the framework and with the support of the European Science Cluster of Astronomy & Particle physics ESFRI research infrastructures (ESCAPE), funded by the European Union's Horizon 2020 - Grant N. 824064.