Deep Learning avancée

Lilian HOLLARD

Master 2 CHPS

lilian.hollard@univ-reims.fr

liciis.univ-reims.fr

ORGANISATION DU COURS

- « Deep dive » Deep Learning
  - Reprise d'élément communs
  - Compréhension en détail du fonctionnement d'un réseau de neurones
  - Convolutional neural networks
    - Bloc de base
    - Différent type d'architecture

Introduction

- Compréhension des architectures de réseaux de neurones en détails
- Construction « from-scratch » – pas de notebook prérempli, désolé ☺
- CM/TD/TP tout mélangé
  - Cours interactif, prenez vos machines !!

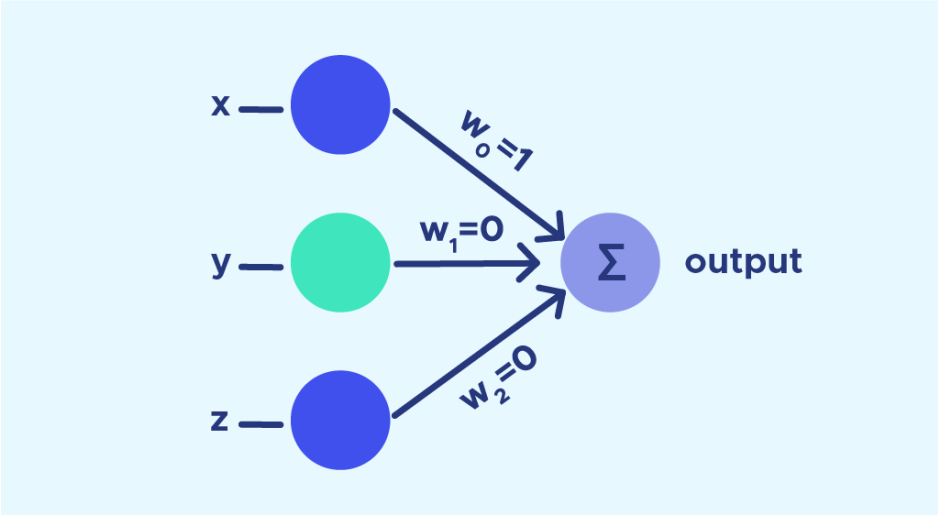
- Objectifs:
  - Première partie : MLP, CNNs, ResNets + entrainement sur CIFAR10
  - Deuxième partie (plus tard dans l'année) : Transformers (architecture tirée des GPTs)
- Notation :
  - Projet à rendre – encore à définir
  - Je prends des notes ! Mais que pour les points bonus ☺

Création d'un premier modèle

- Multi-layer perceptron
  - Connu, mais on va essayer de comprendre pourquoi et comment ça marche.

    - $y = x\,W^T + b$

  - Un perceptron, c'est un produit scalaire, comme ce qui est présenté ci-dessus



Création d'un premier modèle

- Multi-layer perceptron
  - Connu, mais on va essayer de comprendre pourquoi et comment ça marche.
    - $y = x\,W^T + b$
  - Un perceptron, c'est un produit scalaire, comme ce qui est présenté ci-dessus
    - C'est-à-dire, que on va depuis une multitude d'entrée, et une multitude de poids de même taille : créer une valeur unique…
    - Pourtant, un mlp en tensorflow ou en pytorch vous retourne plusieurs valeurs ?
      - C'est à ça que sert la dimension de sortie

Création d'un premier modèle

- Multi-layer perceptron
  - PyTorch implémentation

```python
import torch
import torch.nn as nn

mlp = nn.Linear()
```

```
Init signature:
nn.Linear(
    in_features: int,
    out_features: int,
    bias: bool = True,
    device=None,
    dtype=None,
) -> None
Docstring:
Applies a linear transformation to the incoming data: :math:`y = xA^T + b`

This module supports :ref:`TensorFloat32<tf32_on_ampere>`.

On certain ROCm devices, when using float16 inputs this module will use
:ref:`different precision<fp16_on_mi200>` for backward.
```

Création d'un premier modèle

- Héritez de la classe **nn.Module** !

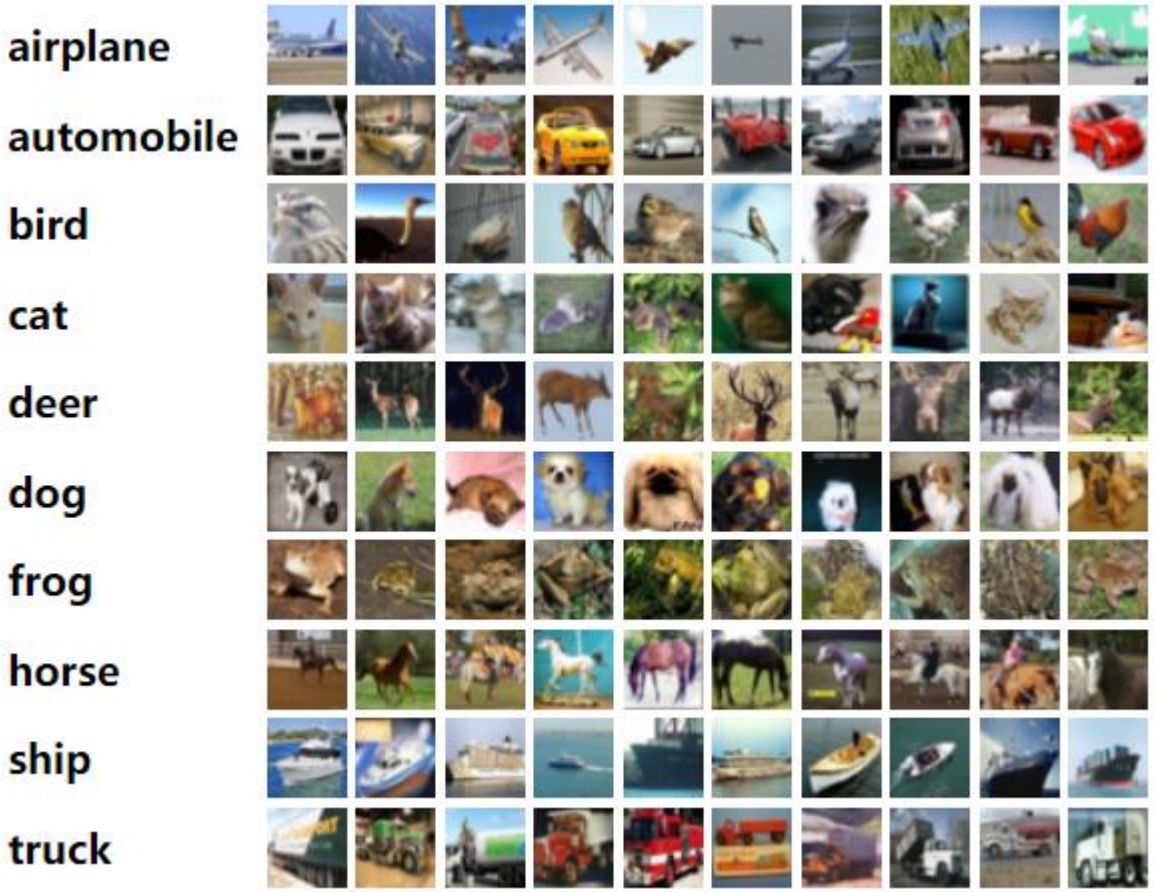
```python
import torch
import torch.nn as nn


class my_class(nn.Module):
    def __init__(self, @param):
        super().__init__()
        #all init
        mlp = nn.Linear(32, 64)
    def forward(self, x):
        #code for forward pass
        return mlp(x)
```

Création d'un premier modèle

- A vous de jouer !
  - https://lilianhollard.github.io/cours/pytorch.html
  - Allez jusqu'à « Entrainer sur CIFAR10 »
    - N'hésitez pas si vous avez des questions !

PRÉSENTATION DU DATASET

- CIFAR10
  - Taille d'image : 32x32x3
  - 10 classes
  - Dataset réputé dans le milieu du Deep Learning



PRÉSENTATION DU DATASET

- En PyTorch, facile à charger

[12]:
```
import torchvision
import torch

import matplotlib.pyplot as plt

import torchvision.transforms as transforms
```

[2]:
```
path = "../datasets/"
```

[39]:
```
train_dataset = torchvision.datasets.CIFAR10(root=path, train=True, download=True, transform=transforms.ToTensor())
test_dataset = torchvision.datasets.CIFAR10(root=path, train=False, download=True, transform=None)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

[8]:
```
plt.imshow(train_dataset[0][0])
```

Entrainer avec PyTorch

- Faire le découpage du jeu de donnée
  - Avec torchvision -> Déjà fait.
- Faire un découpage par « batch »
  - **train = torch.utils.data.DataLoader(...)**
  - **test = torch.utils.data.DataLoader(...)**

- Récupérer les valeurs de chaque batchs pour l'entrainement
  - for batch in train:
    images, labels = batch

Entrainer avec PyTorch

- Retenez bien cette boucle
  - En PyTorch, pas de .fit() comme en tensorflow qui cache absolument tout

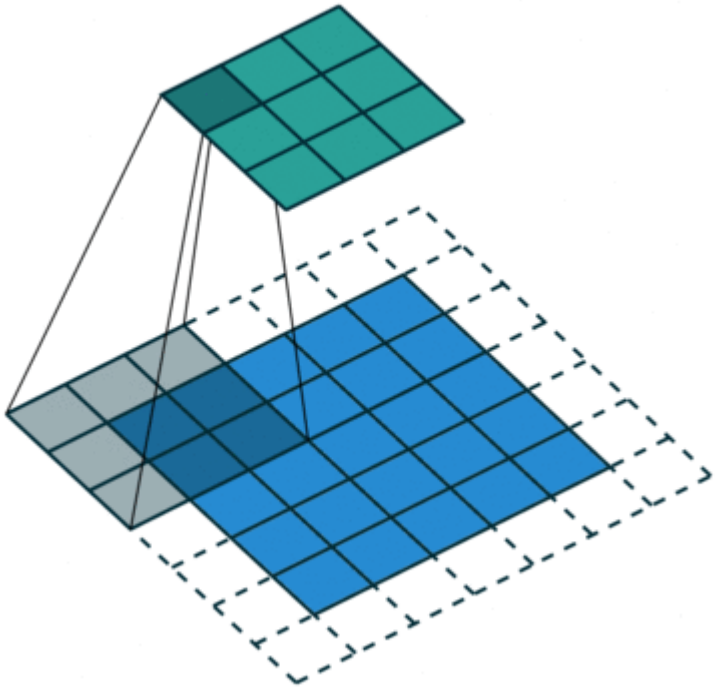
  - Boucle d'entrainement:
    - 1. Forward pass : y_pred = model(images)
    - 2. Calculez la « loss » : loss = function_de_loss(y_pred, labels)
    - 3. Gradient à zero : optimizer.zero_grad()
    - 4. Backpropagation à l'aide du résultat de la fonction de loss : loss.backward()
    - 5. Faire la descente de gradient : optimizer.step()

Entrainer avec PyTorch

- A vous de jouer !
  - https://lilianhollard.github.io/cours/pytorch.html
  - Finissez l'intégralité du chapitre 1.
    - N'hésitez pas si vous avez des questions !

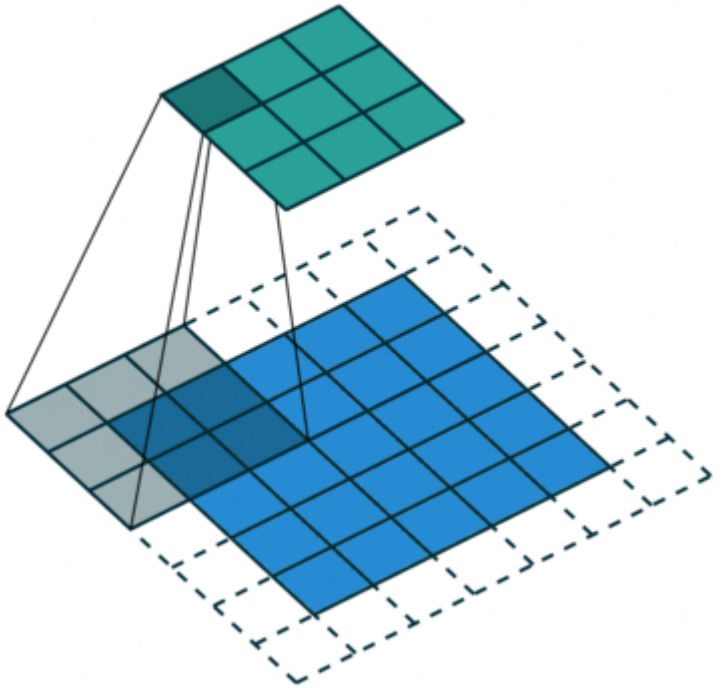
Qu'est-ce qu'un réseau de neurones

- Convolutional neural network



Qu'est-ce qu'un réseau de neurones

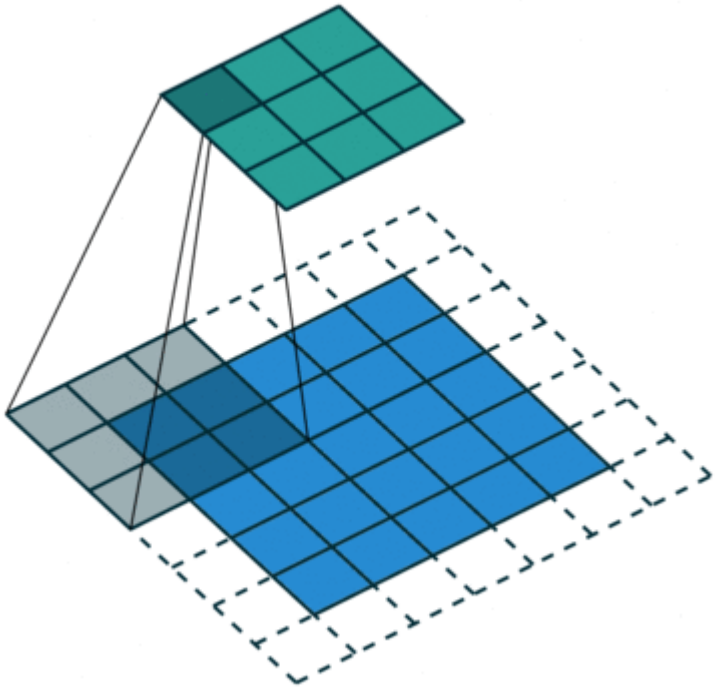
- Convolutional neural network
  - A quoi ça sert ?
    - Créer une fonction capable de répondre (et généraliser) la solution d'un problème



Qu'est-ce qu'un réseau de neurones

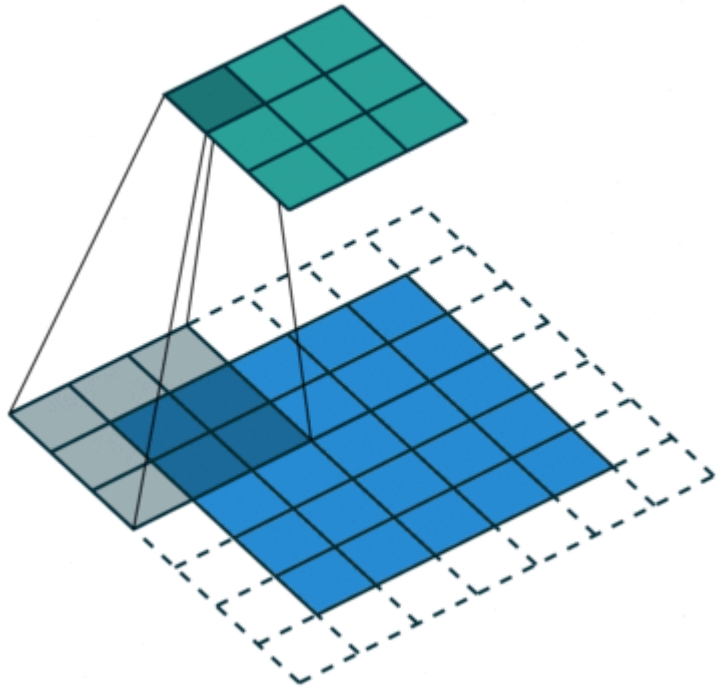
- Convolutional neural network
  - A quoi ça sert ?
    - Créer une fonction capable de répondre (et généraliser) la solution d'un problème
  - En mathématiques :
    - Système linéaire : résoudre *n* équations nécessites *n inconnus*

$$2x + 3y = 20$$
$$4x - 2y = 12$$

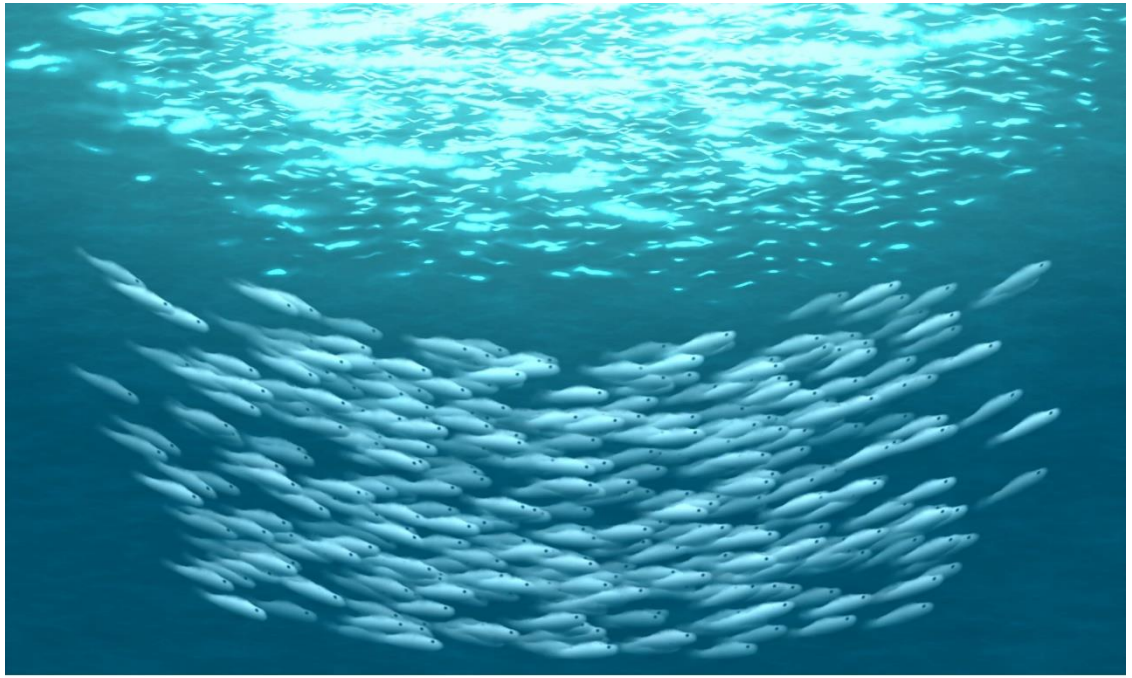


Qu'est-ce qu'un réseau de neurones

- Convolutional neural network
  - A quoi ça sert ?
    - Créer une fonction capable de répondre (et généraliser) la solution d'un problème
  - En vision par ordinateur…
    - C'est plus compliqué !



QU'EST-CE QU'UN RÉSEAU DE NEURONES



School of Fish by Hamid Naderi Yeganeh

The above picture is a converted version of a 2000×1200 image. For $m = 1, 2, 3, \ldots, 2000$ and $n = 1, 2, 3, \ldots, 1200$, the color of the pixel of the row n and the column m is

$$\mathrm{rgb}\left(F\left(H_0\left(\tfrac{m-1000}{600}, \tfrac{601-n}{600}\right)\right), F\left(H_1\left(\tfrac{m-1000}{600}, \tfrac{601-n}{600}\right)\right), F\left(H_2\left(\tfrac{m-1000}{600}, \tfrac{601-n}{600}\right)\right)\right), \text{ where } F(x) = \left\lfloor 255 e^{-e^{-1000x}} |x|^{e^{-e^{1000(x-1)}}} \right\rfloor, \text{ and}$$

$$H_v(x,y) = \frac{1}{200}\left(8 - 2(2-v)^2 + (-5v^2 + 11v + 14)e^{-e^{\frac{3}{20}v^2 + \frac{3}{125}(5y-6)^2 - \frac{3}{40}}}\right)\left(30 - 20A_0(x,y) + 3\left(1 - A_1(x,y)\right)\right) + U(x,y),$$

$$U(x,y) = \sum_{s=1}^{50} \frac{80-s}{80}\left(\frac{2}{5} - \frac{1}{2}\sin\left(2C_s(x,y)\right)\right) e^{-e^{200\sin^{18}\left(B_s(x,y) - \frac{41}{100}\right)\sin^2\left(C_s(x,y) - \frac{1}{5}\right) - 199}} J_s(x,y) \prod_{u=0}^{s-1} 1 - e^{e^{-1000\left(u - \frac{1}{2}\right)}} J_u(x,y),$$

$$J_s(x,y) = e^{-e^{1000\left(|B_s(x,y)| - 2\pi\right) - e^{1000\left(|C_s(x,y)| - \pi\right) - e^{\frac{397R_s(x,y) - 400}{10}\sin^2(B_s(x,y))|\sin(C_s(x,y))|^{\left(\frac{6}{5} + \sin\left(2H_s(x,y)\right)\right)\left(5 - 4R_s(x,y)\right) - \frac{7}{10}\frac{397R_s(x,y) - 400}{10} + \frac{397R_s(x,y) - 400}{10}\frac{1+2R_s(x,y)}{50}\left(E_0(x,y) + E_1(x,y)\right)}}},$$

$$R_s(x,y) = e^{-e^{-40\left(\left|\sin\left(B_s(x,y) + \frac{67}{100}\right)\right| - \frac{19}{20}\right)}} + \frac{9}{10}e^{-e^{-40\left(\left|\sin\left(B_s(x,y)\right)\right| - \frac{19}{20}\right)}}, \quad C_s(x,y) = 20P_s(x,y) + \frac{2}{5}\cos(24s)\, e^{-e^{-1000\sin\left(B_s(x,y)\right)}}, \quad B_s(x,y) = 6Q_s(x,y) + 2\cos(72s)\, e^{-e^{-1000\sin\left(10P_s(x,y)\right)}},$$

$$P_s(x,y) = y + \frac{1}{2} - \cos(9s)\frac{x}{10} - \frac{x^2}{5} + \frac{7}{20}\cos(39s), \qquad Q_s(x,y) = x + \frac{3}{10}\cos(79s),$$

$$A_v(x,y) = \prod_{s=1}^{50} e^{-e^{(50-40v)\left(\cos^2\left(\frac{21^3}{20^3}\left(\frac{3}{2} + \frac{\cos(72s)}{2}\right)\right)\left(x + \frac{3}{10}\cos\left((10+3\cos(s))y + 3s\right)\right) + \cos\left(\frac{1}{2}\frac{21^2}{20^3}x + 14s\right) + 2\cos(24s)\right)\cos^2\left(\frac{21^3}{20^3}(8+2\cos(4s))\left(y + \frac{7}{1000}\cos\left((20+3\cos(8s))x + 3s\right) + \frac{1}{250}\cos\left((70+20\cos(8s))x + \cos(50x+2) + 2\cos(7s)\right)\right) + 2\cos(16s)\right) + L_{v,s}(x,y)}},$$

$$L_{v,s}(x,y) = -\frac{6}{5} + \left(\frac{1}{4} + \frac{v}{5}\right)K_s(x,y) + \frac{3}{100}E_0(x,y) + \frac{3}{100}E_1(x,y), \quad K_s(x,y) = e^{-e^{20\left(\left(x + \frac{3}{10}\cos(27s)\right)^2 + 5\left(y - 1 + \frac{1}{5}\cos(37s)\right)\left(y + \frac{1}{5}\cos(5x+2s) - 1 + \frac{1}{5}\cos(37s)\right) - 1 - \frac{s}{100} + \frac{3}{4}\cos(8s)\right)}}, \quad E_v(x,y) = \sum_{s=1}^{50} W_{v,s}(x,y),$$

$$W_{v,s}(x,y) = \frac{3+57v}{200}(23-2v)^{-s}20^s\cos\left(8(14-3v)^s10^{-s}(1+3\cos(10s))(\cos(2s^2)\,x + \sin(2s^2)\,y) + 4\cos\left((14-3v)^s10^{-s}(\cos(7s^2)\,x + \sin(7s^2)\,y)\right) + 2\cos(5s)\right)$$
$$\times\cos\left(8(14-3v)^s10^{-s}(1+3\cos(10s))(\cos(2s^2)\,y - \sin(2s^2)\,x) + 4\cos\left((14-3v)^s10^{-s}(\cos(8s^2)\,x + \sin(8s^2)\,y)\right) + 2\cos(5s)\right).$$

Qu'est-ce qu'un réseau de neurones

- Convolutional neural network
  - Pourquoi les CNNs ?
    - 1. Les MLPs sont trop couteux et ne permettent pas de donner une vision spatiale
    - Premier « TP »