RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

METEO FRANCE
À VOS CÔTÉS, DANS UN CLIMAT QUI CHANGE

Retour sur le TP n°1 et introduction au TP n°2

Pierre Lepetit
ENM, le 17/10/2025

Retour sur le TP1 :

Notions abordées et notions à creuser

- Perceptron à une couche pour la **classification** :
  Code objet ; Initialisation des poids (He/Xavier)
  Renvoie un vecteur de probabilité → ajout d'une règle de décision (torch.max(.)[1])

- Pytorch Dataset :
  Code objet (__getitem__ ;__len_) ;
  Sélection d'un élément ; Augmentation de données

- PyTorch DataLoader :
  Code (arguments : batch size, shuffle) ;
  pondération du tirage via l'utilisation d'un sampler

Retour sur le TP1 :

Notions abordées et notions à creuser

- NLLL (Negative Log Likelihood Loss) pour la **classification**:
  Formule ($-\ln(p_c)$);
  Pourquoi pas |torch.max(f(x))[1] – c| ? Pourquoi « Likelihood » ?
  CrossEntropyLoss, BinaryCrossEntropy

- Descente de gradient stochastique (**SGD**) par mini-batches :
  Pourquoi par mini-batches ? Comment est-elle paramétrée ?
  Rôle de loss.backward() ; Comment récupérer les gradients ?
  En arrière-plan ? (Autograd Computational Graph → torchviz.make_dot())

- Perceptron multicouche et séparation non-linéaire (XOR)
  Classification multiclasse ; Softmax function ; one hot encoding
- Notion de test, de performances en généralisations

TP n°2 : Réseaux convolutifs (Convolutional Neural Networks)

Deux périodes clefs :

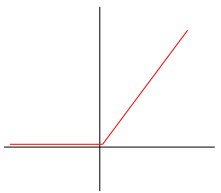
- Premiers développements/applications : 1989 – 1995
  Réseaux peu profonds, moins de 100.000 paramètres, sur CPU.
  - » Lecture de codes postaux
  - » Reconnaissance de phonèmes (signaux 1D)
  - » Détection de fausses signatures (signaux 1D, réseaux « siamois »)

- 2012 – 2017 :
  Réseaux profonds, plus de 10 M. de paramètres, sur GPU
  - » AlexNet → +ReLU activations
  - » VGG, ResNet → +Initialisation + Régularisation (Dropout +BatchNorm)
  - » FCN (Fully Convolutional Network) → prédiction « dense »
  - » Bibliothèques ad hoc (Tensorflow, PyTorch)

TP n°2 : Réseaux convolutifs (Convolutional Neural Networks)

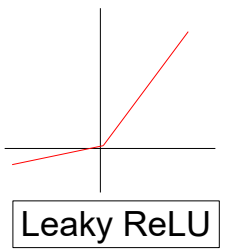
Qu'est-ce que c'est ?

- Idée de base : invariance spatiale / weight sharing
- Un « neurone » = un « noyau de convolution » + un biais.
  - » Représentation visuelle
  - » Formellement
  - » Implémentation en arrière-plan (éléments théoriques)



- Une couche de convolution = Conv2d (ou Conv1d, Conv3d) + activation funct.

```
self.conv = nn.Conv2d(3, 64, kernel_size=7, padding=3, stride=2, bias=True)
self.relu = nn.ReLU()
```



- Un CNN complet = Empilement de :
  - couches de convolution
  - opérations de « Pooling » → réduction des dimensions spatiales
  - couches finales : dépendent de la tâche (e.g. perceptron pour classif.)

TP n°2 : Réseaux convolutifs (Convolutional Neural Networks)

Objectif **principal** du TP = décrypter ces figures :