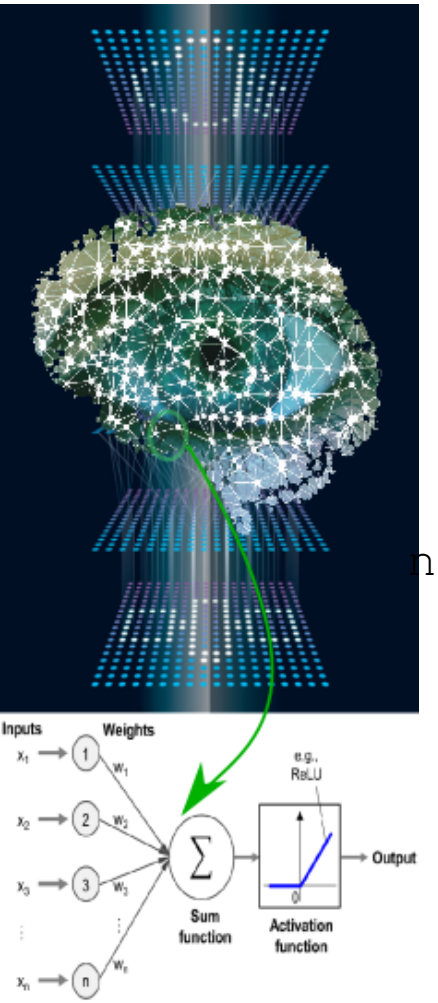
Réseau de neurones (II)



Cours 4
ML Master DAC

Nicolas Baskiotis
nicolas.baskiotis@sorbonne-universite.fr

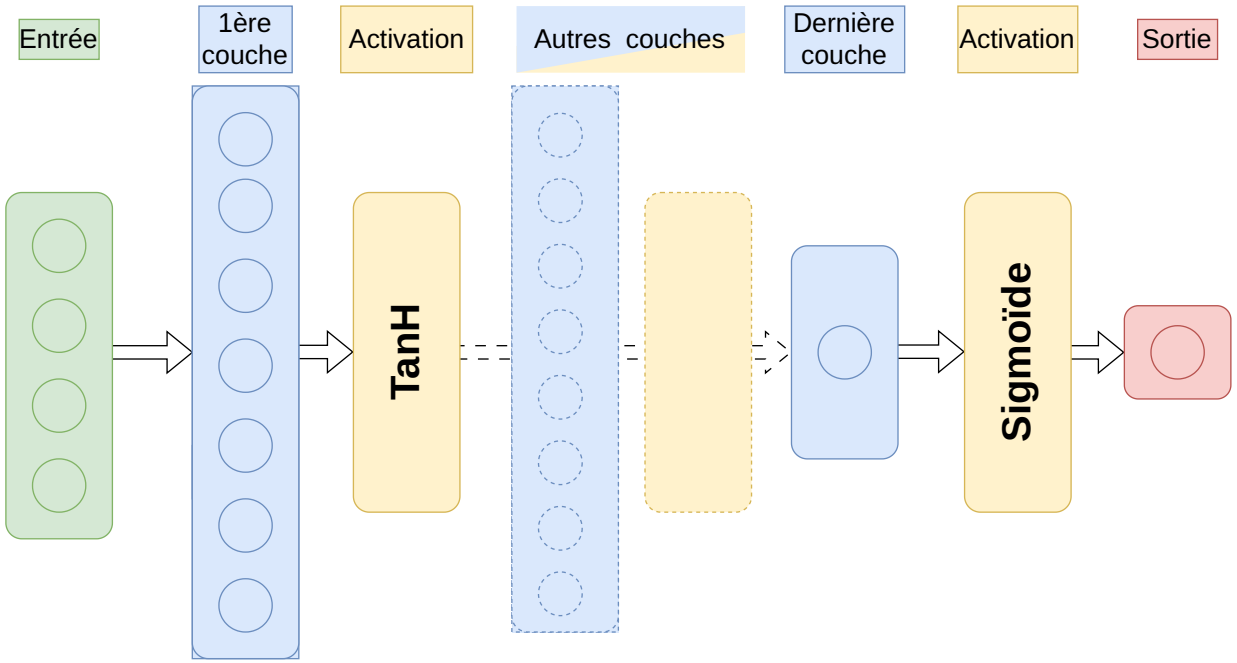
équipe MLIA,
Institut des Systèmes Intelligents et de Robotique (ISIR)
Sorbonne Université

S2 (2023-2024)

Plan

Réseau MLP : Une architecture passe-partout

Un réseau MLP est une succession de couches linéaires et de fonctions d'activation

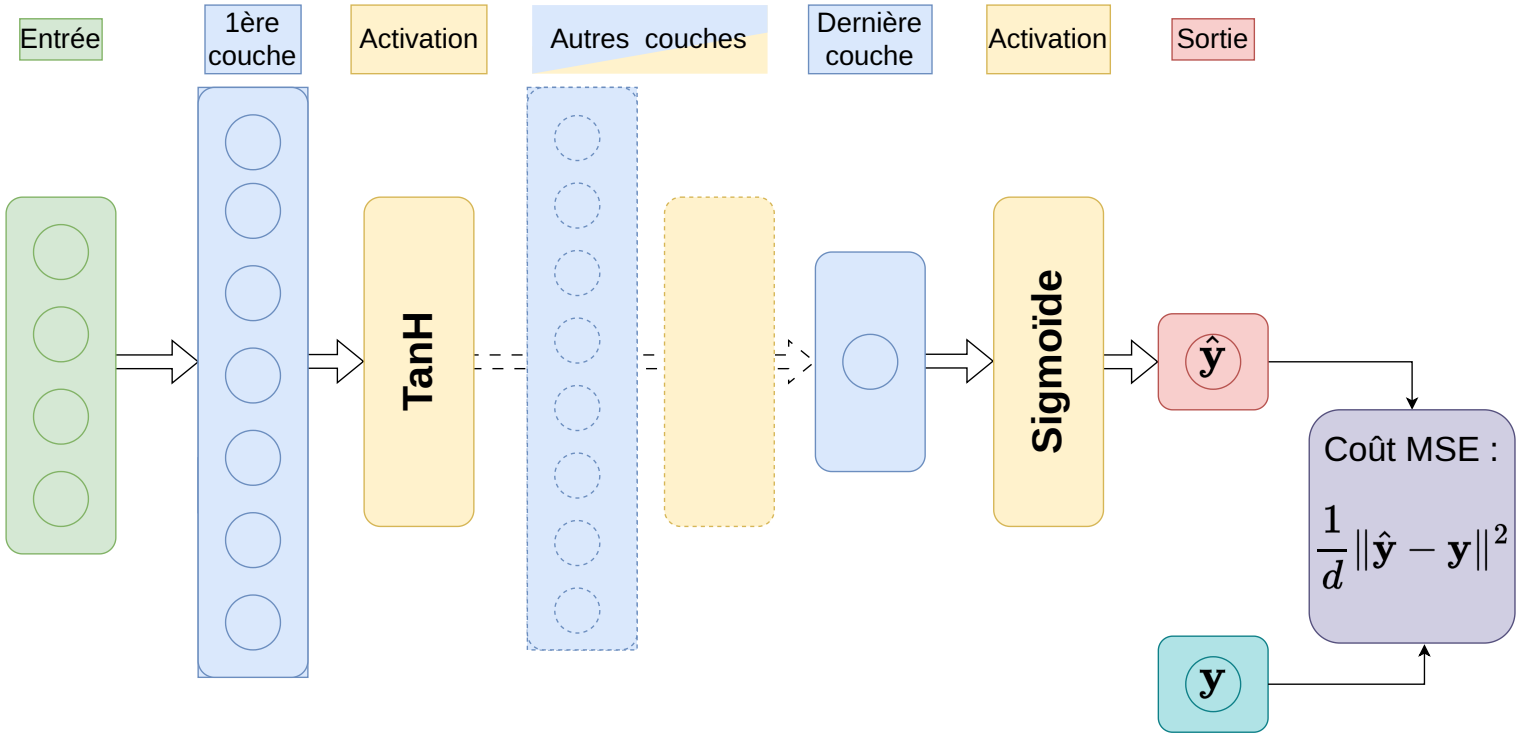


Propriétés du Multi-Layer Perceptron (MLP)

- Idéal pour les tâches simples (régression, classification, multi-classes . . .)
- Architecture que l'on retrouve quasiment dans toutes les autres architectures
- Mais non adaptée sur des entrées complexes (texte, image)
- Très sujet au sur-apprentissage avec l'augmentation du nombre de couches

$\Rightarrow$ **La généricité de l'architecture : une histoire de coût !**

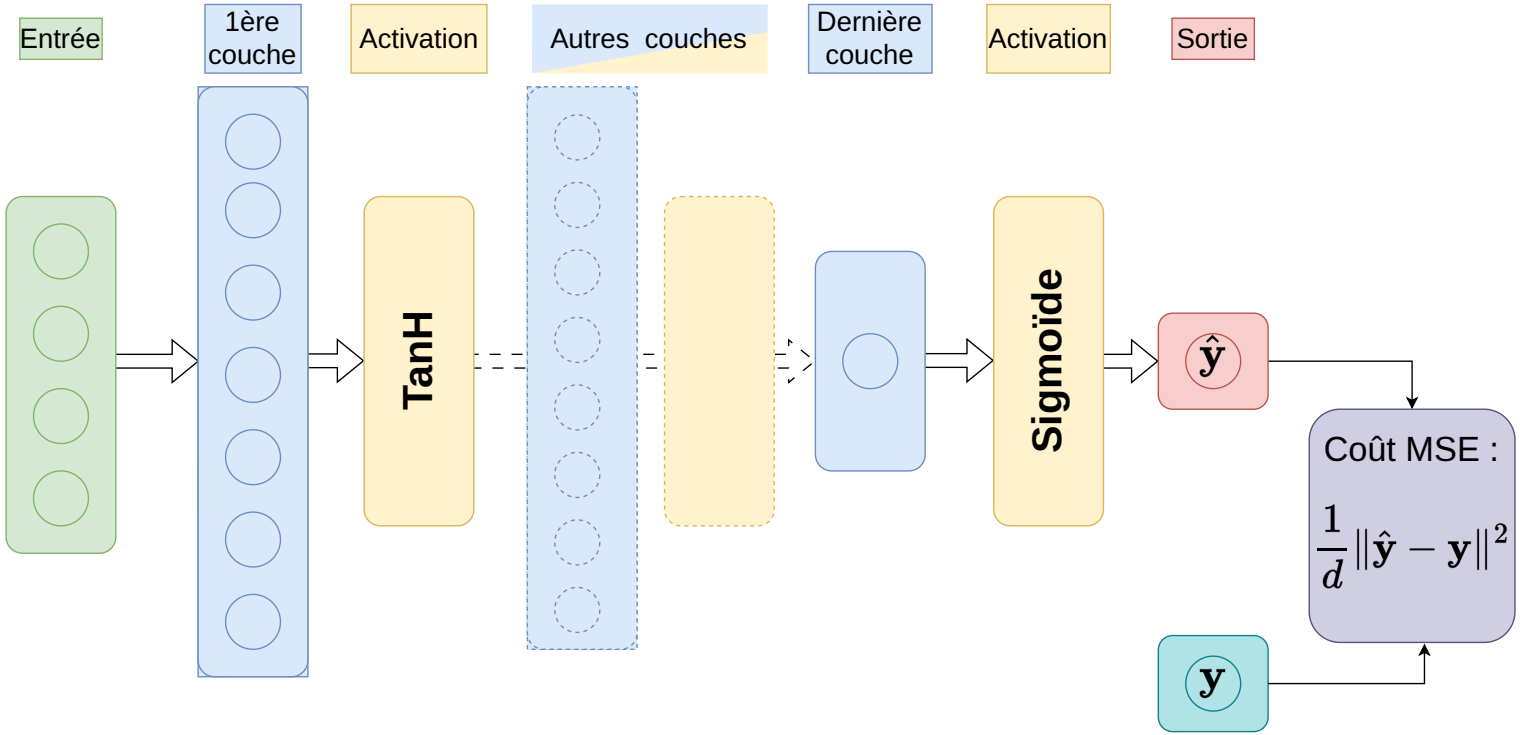
Régression : coût MSE



Coût Moindres Carrés

- Usuel dans le contexte de la régression
- Adapté également aux sorties multi-variées
- Attention : choisir une activation de sortie adaptée !
  - si normalisation entre 0 et 1 $\rightarrow$ sigmoïde
  - si normalisation entre -1 et 1 $\rightarrow$ tangente hyperbolique

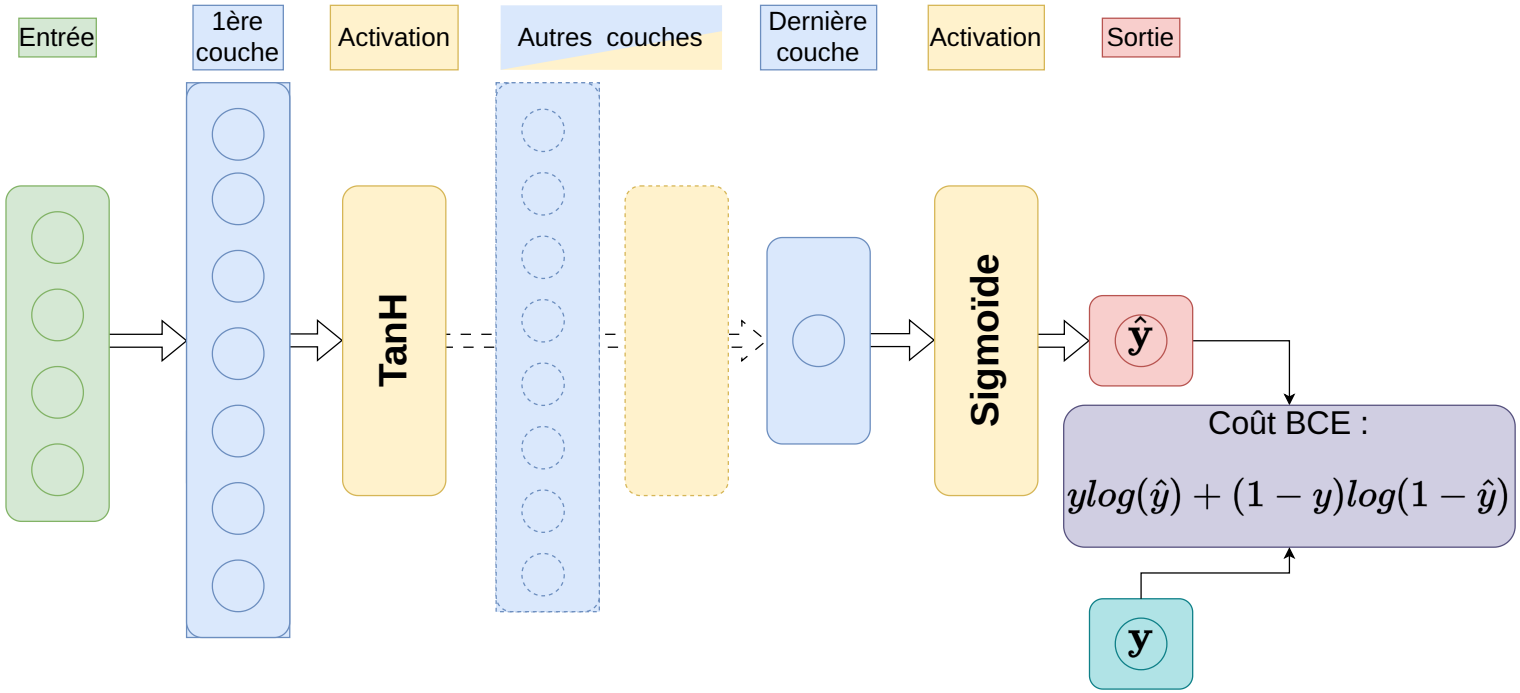
Classification binaire (1) : coût MSE



La MSE fonctionne également !

- Codage soit $\{0, 1\}$ des classes, soit $\{-1, 1\}$
- Seuil à 0.5 pour départager entre les deux classes (ou autre pour régler le compromis rappel/précision)
- Mais pas idéal, le coût n'est pas assez "piqué" vers les extrêmes

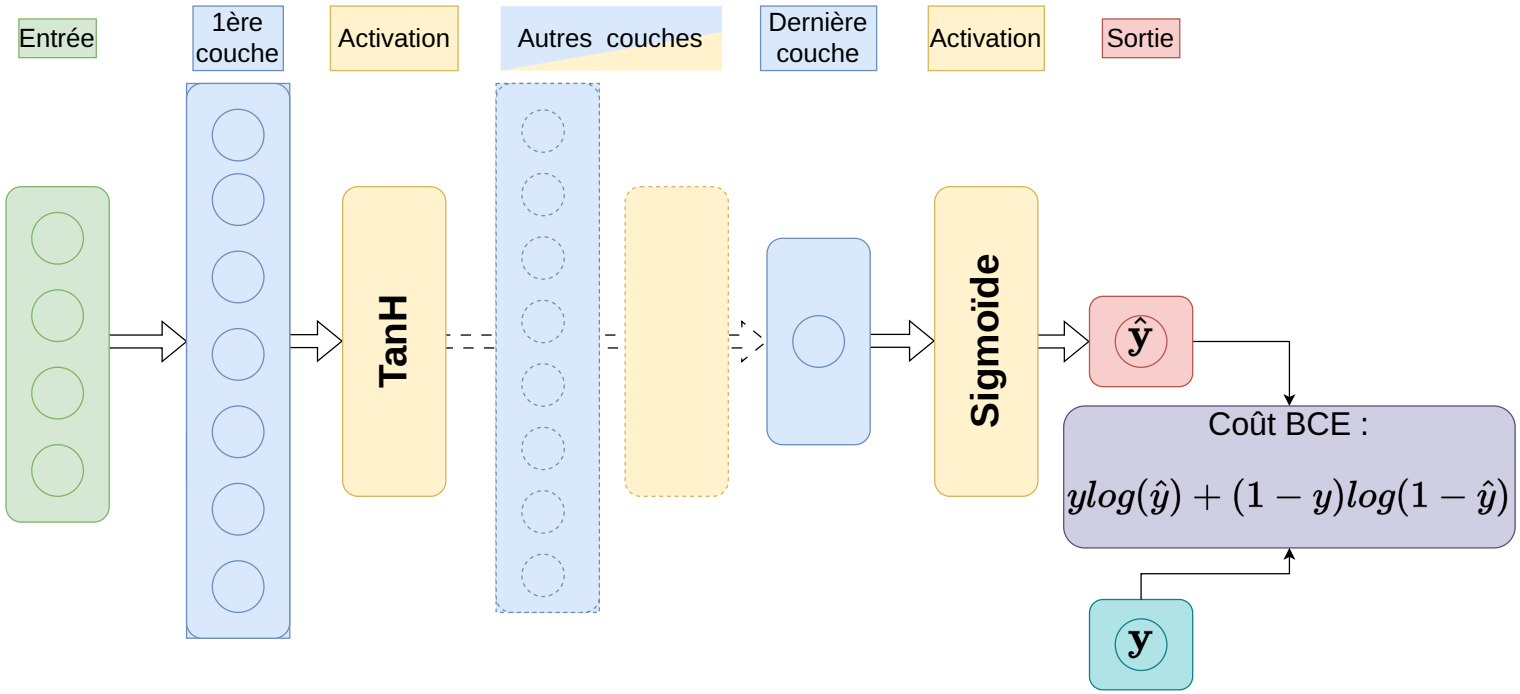
Classification binaire (2) : coût BCE



Binary Cross Entropy

- Codage soit $\{0, 1\}$ des classes
- Sigmoïde comme dernière couche d'activation (la sortie représente la probabilité de la classe 1)
- Seuil à 0.5 pour départager entre les deux classes (ou autre pour régler le compromis rappel/précision)
- Encourage les valeurs extrèmes 0 et 1.

Classification binaire (2) : coût BCE



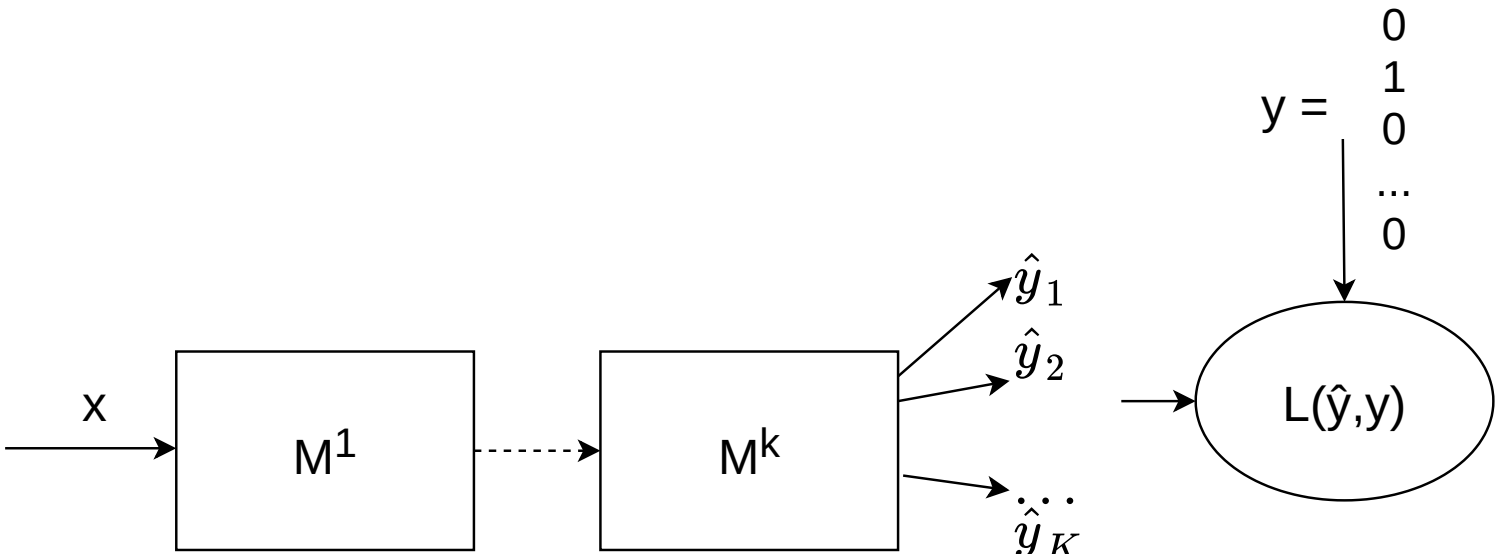
Binary Cross Quoi ?

- Hypothèse : distribution binomiale sur la sortie :
- Soit $\hat{y} = f_\theta(\mathbf{x})$ la sortie du réseau, qui modélise $p(y=1|\mathbf{x}) = f^*(\mathbf{x})$

$\Rightarrow$ $P(y|\mathbf{x}) = f^*(\mathbf{x})^y(1-f^*(\mathbf{x}))^{1-y}$

- Neg-Log-vraisemblance :
  $NLL((\mathbf{x},y);\theta) = -ylog(f_\theta(\mathbf{x})) - (1-y)(1-f_\theta(\mathbf{x})) = -ylog(\hat{y}) - (1-y)log(\hat{y})$
- Equivalent Cross-Entropie : $H(p,q) = -\sum_{y\in\mathcal{Y}} p(y)log(q(y))$, avec p la distribution des "vrais" labels et q la distribution des labels prédits

Multi-classe



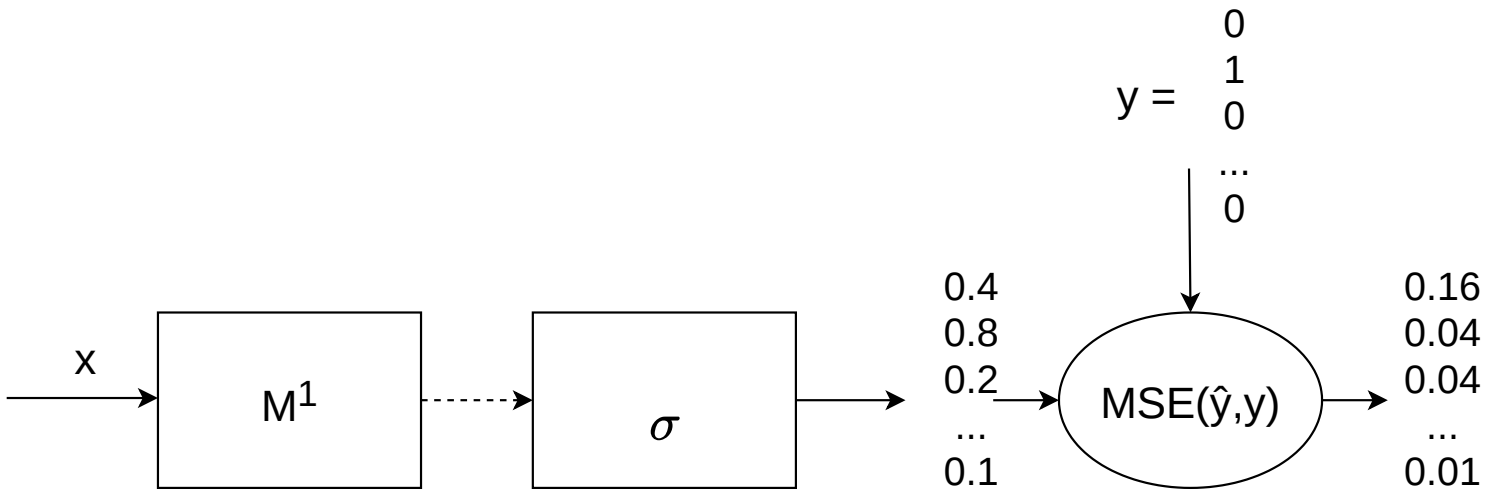
Quand il faut prédire K classes

- K sorties
- Utilisation de vecteurs 1-hot pour la supervision:

$$\mathbf{y} = (0, 0, \ldots, 1, \ldots, 0)$$

avec $y_i = 0$ pour i différent de la bonne classe,
$y_k = 1$ pour k l'indice de la bonne classe.

Utilisation de la MSE



Fonction de coût problématique

- Sortie du réseau entre 0 et 1 $\Rightarrow$ utilisation d'une sigmoïde
- Mais :
  - La similarité au vecteur de sortie n'est pas le plus important, c'est l'argmax qui nous intéresse le plus
  - Pas plus d'efforts mis sur la maximisation de la sortie de la bonne classe que la minimisation des autres sorties

Coût Cross-entropique

SoftMax : Transformer les sorties en distribution

$$\text{SoftMax}(z)_i = e^{z_i} / \left(\sum_{j=1}^{K} e^{z_j} \right), \quad \sum_{i=1}^{K} \text{SoftMax}(z)_i = 1$$

Coût Cross-entropique Multi-Classe

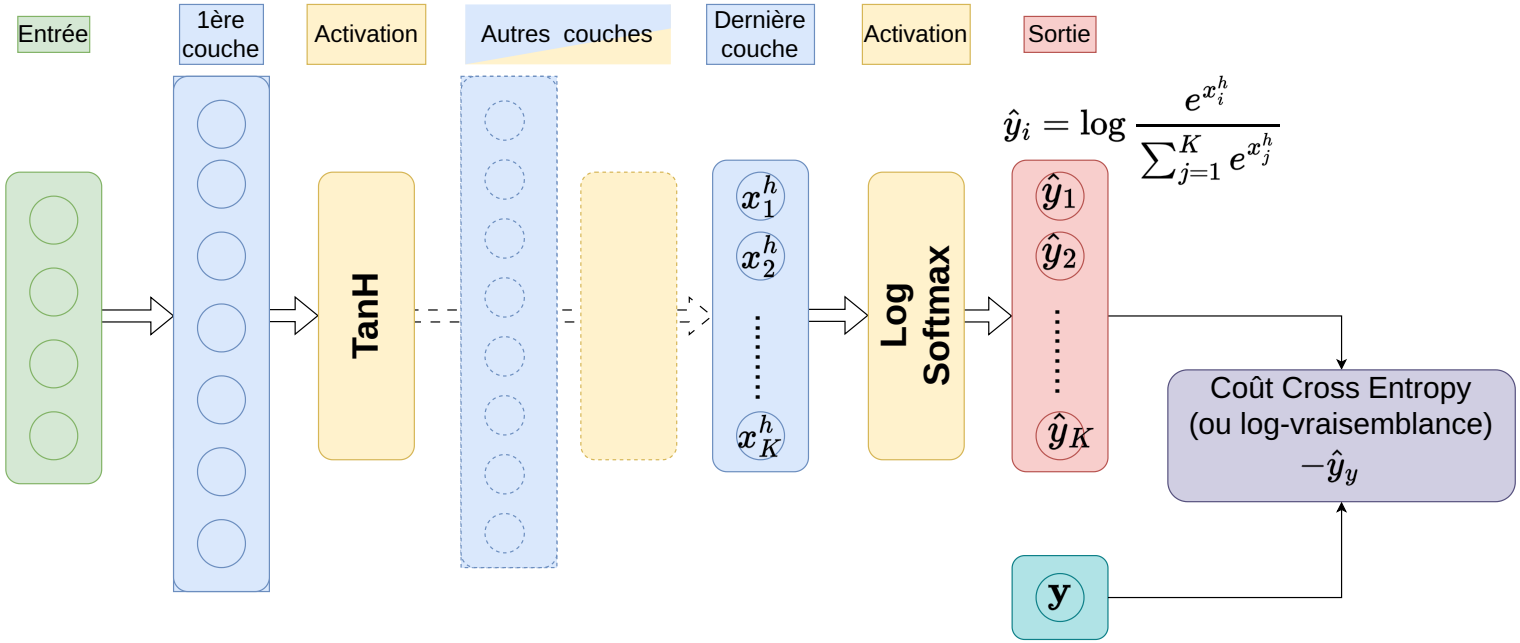
- $CE(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$
- Dans le cas où $\mathbf{y}$ est un vecteur one-hot de la classe k : $CE(\mathbf{y}, \hat{\mathbf{y}}) = -\log(\hat{y}_k)$
- Identique à la log-vraisemblance et une hypothèse multinomiale sur les sorties
- Combinaison SoftMax et Cross-entropie :

$$CE(\mathbf{y}, SoftMax(z)) = -z_k + \log \left(\sum_{j=1}^{K} e^{z_j} \right), \quad \frac{\partial CE(\mathbf{y}, SoftMax(z))_i}{\partial z_i} = Softmax(z)_i - 1_{i=k}$$

Cross-entropie binaire

Pour le multi-class multi-label en particulier, cross-entropie sur chaque sortie de manière indépendante: $BCE(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{K} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)$

Classification multi-classes : coût Cross Entropy



Problème à K classes

- K sorties, chacune représente la probabilité de la classe correspondante
- Softmax sur les sorties pour obtenir une distribution de probabilité
- La classe prédite correspond à l'index de la sortie la plus grande
- Coût Cross Entropy (équivalent log-vraisemblance) pour l'apprentissage
- Si multi-labels, alors utilisation de la BCE sur chaque sortie (hypothèse de sorties indépendantes les unes des autres)

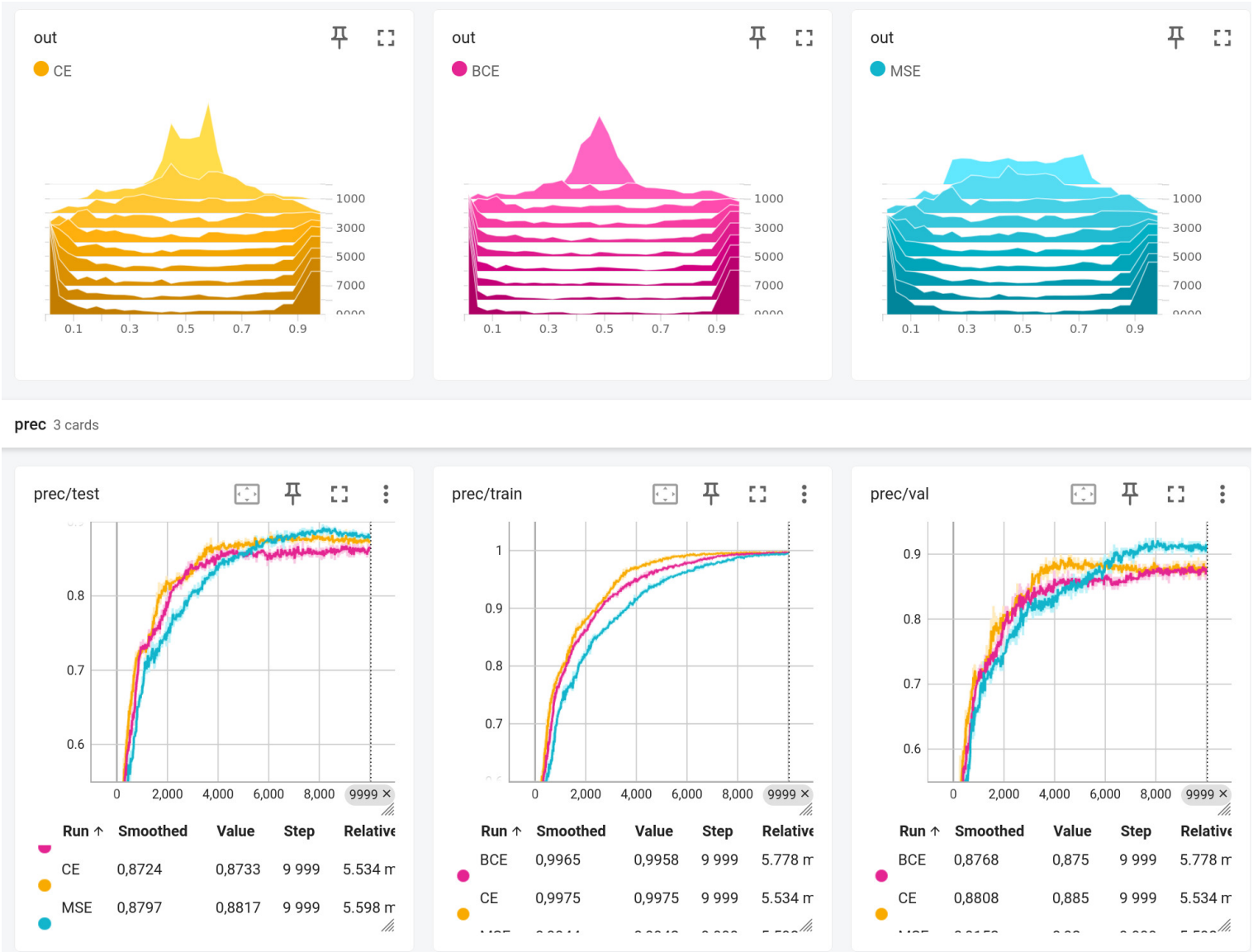
En résumé

La fonction de coût

- Essentielle au bon apprentissage, énormément de variantes
- Tâche-dépendante, données-dépendante, très flexible (pondération des classes par exemple pour les données déséquilibrées)

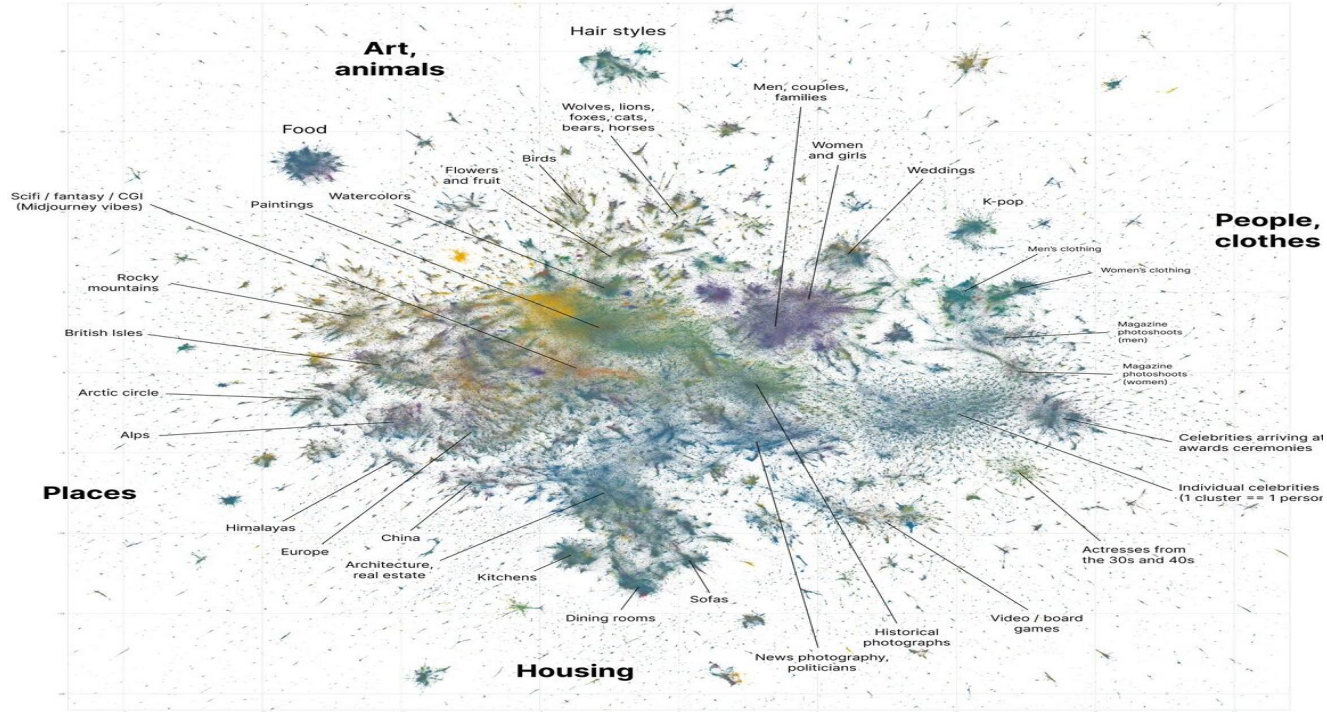
`nn.L1Loss`	error (MAE) between each element in the input x and target y.	`nn.GaussianNLLLoss`	Gaussian negative log likelihood loss.	`nn.MultiLabelMarginLoss`	classification input x (a 2D is a 2D *Tenso*
`nn.MSELoss`	Creates a criterion that measures the mean squared error (squared L2 norm) between each element in the input x and target y.	`nn.KLDivLoss`	The Kullback-Leibler divergence loss.	`nn.HuberLoss`	Creates a crit absolute elen delta-scaled L
`nn.CrossEntropyLoss`	This criterion computes the cross entropy loss between input logits and target.	`nn.BCELoss`	Creates a criterion that measures the Binary Cross Entropy between the target and the input probabilities	`nn.SmoothL1Loss`	Creates a crit absolute elen term otherwi
`nn.CTCLoss`	The Connectionist Temporal Classification loss.	`nn.BCEWithLogitsLoss`	This loss combines a *Sigmoid* layer and the *BCELoss* in one single class.	`nn.SoftMarginLoss`	Creates a crit classification target tensor
`nn.NLLLoss`	The negative log likelihood loss.	`nn.MarginRankingLoss`	Creates a criterion that measures the loss given inputs $x1$, $x2$, two 1D mini-batch or 0D *Tensors*, and a label 1D mini-batch or 0D *Tensor* y (containing 1 or -1).	`nn.MultiLabelSoftMarginLoss`	Creates a crit versus-all loss
`nn.PoissonNLLLoss`	Negative log likelihood loss with Poisson distribution of target.	`nn.HingeEmbeddingLoss`	Measures the loss given an input tensor x and a labels tensor y (containing 1 or -1).		

Exemple : Echequier



Plan

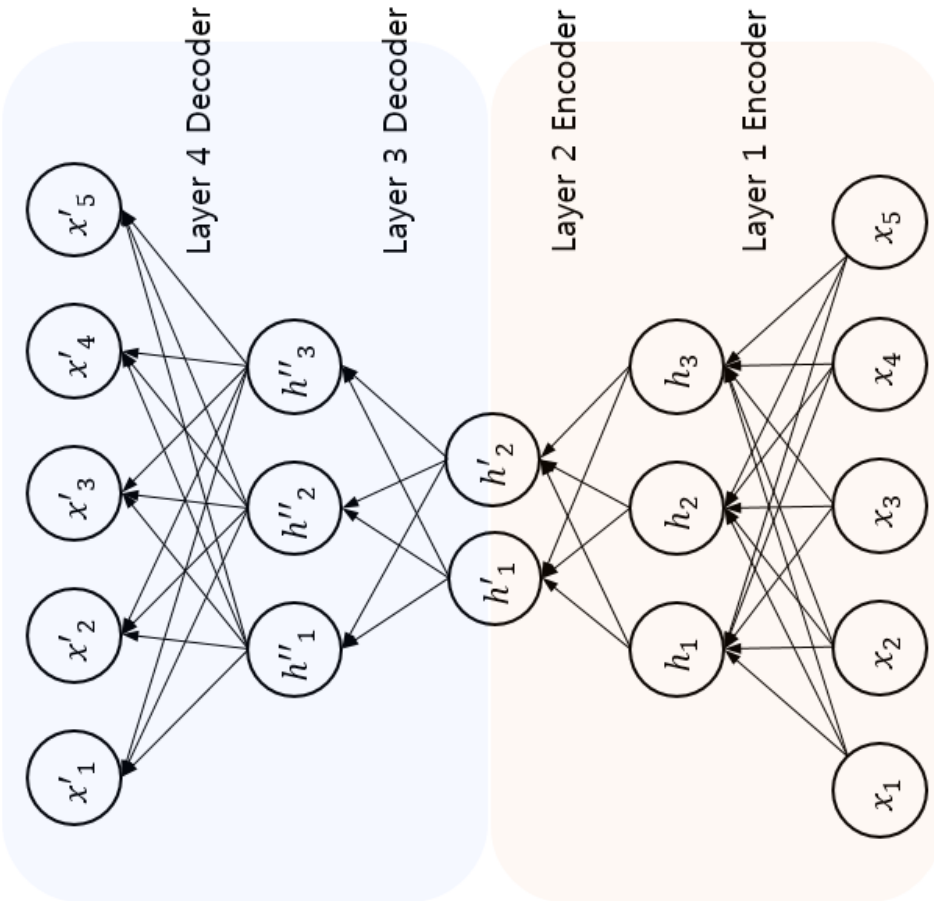
Quand on n'a pas de features ...



Il faut pouvoir projeter des objets discrets dans un espace continu

- Projection aléatoire : a existé (feature hashing), peu efficace
- Pas de description des objets, mais il existe des relations entre eux !
- Un "bon" espace de représentation doit pouvoir exprimer la sémantique latente de ces relations
- Peut être guidé par l'expert (avant le Deep), la tâche (apprentissage end-to-end) ou par les données sans supervision (self-supervised par exemple)

Exemple (1) : Auto-encoders



Apprentissage non-supervisé

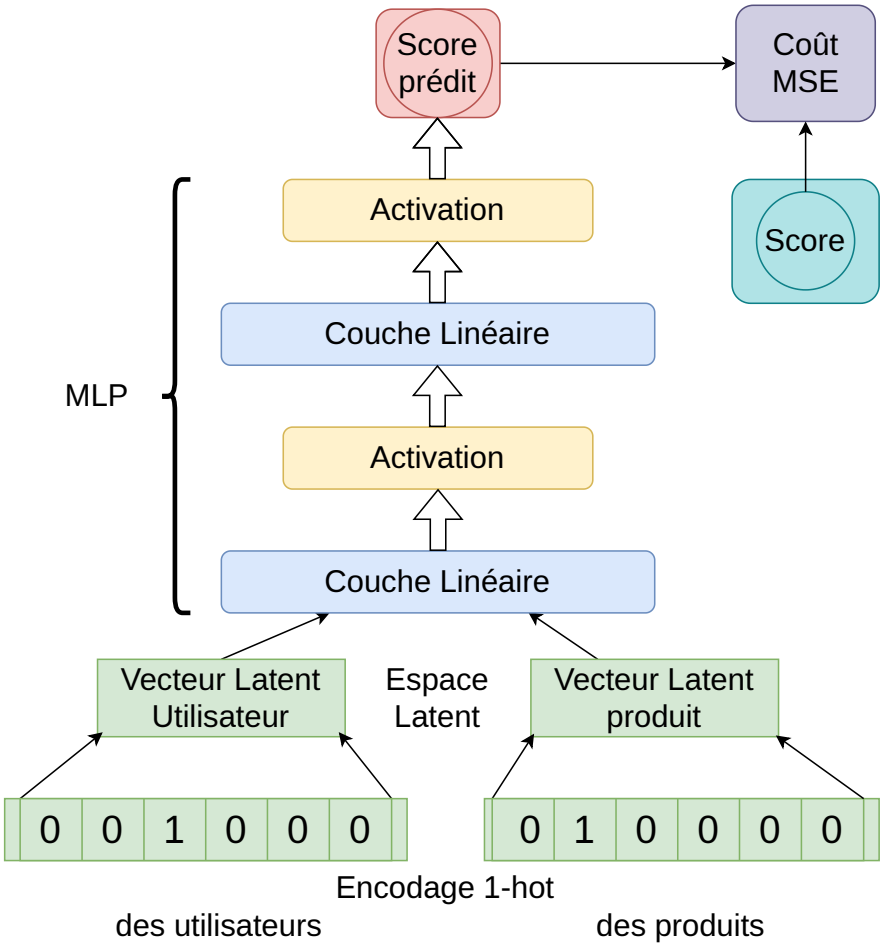
Objectif :

- apprendre une “compression” des données utile pour l’apprentissage
- $f^{-1}(f(x)) \approx x$: la sortie doit être proche de l’entrée

Intérêts :

- Représentation plus adéquate
- clustering des données
- lissage/débruitage
- visualisation
- ...

Exemple (2) : Représentation et recommandation



Données

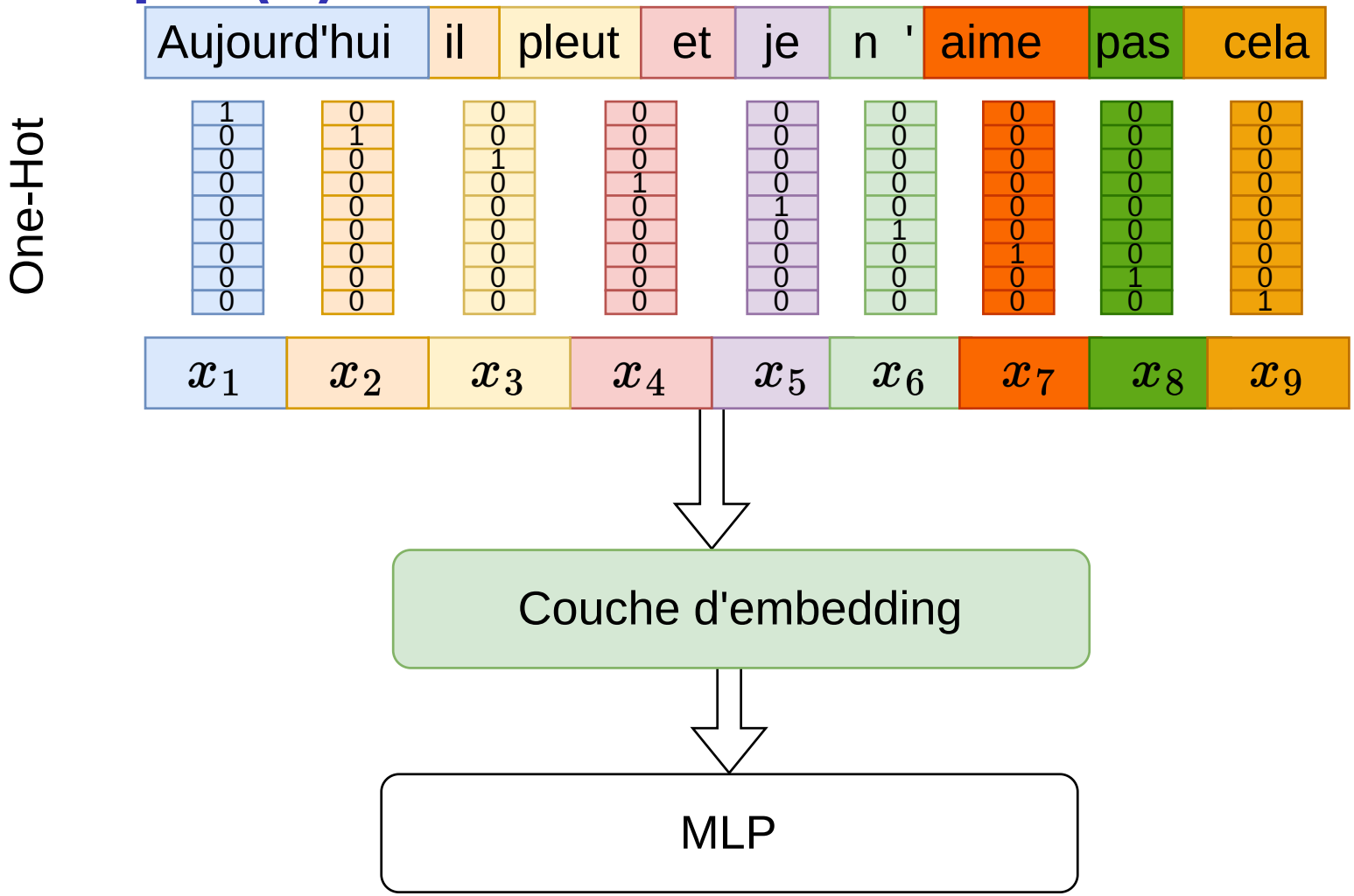
- Des produits (sans description)
- Des utilisateurs (sans description)
- Des notes des utilisateurs sur les produits

Objectif : prédire les avis des utilisateurs

Procédure :

- Encodage one-hot des produits et des utilisateurs (un produit et un utilisateur par dimension)
- Représentation aléatoire initialement
- La back-propagation corrige le MLP et les représentations utilisateurs/produits

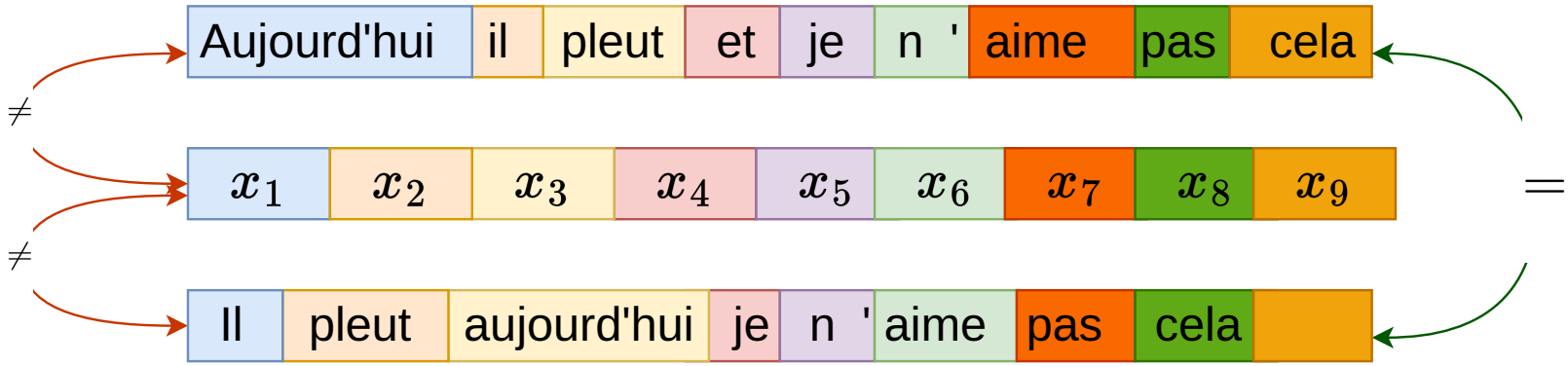
Exemple (3) : le texte



Même objectif :

- Trouver une représentation ”utile” pour les éléments atomiques (token)
- Problème de la tokenisation

Exemple (3) : le texte



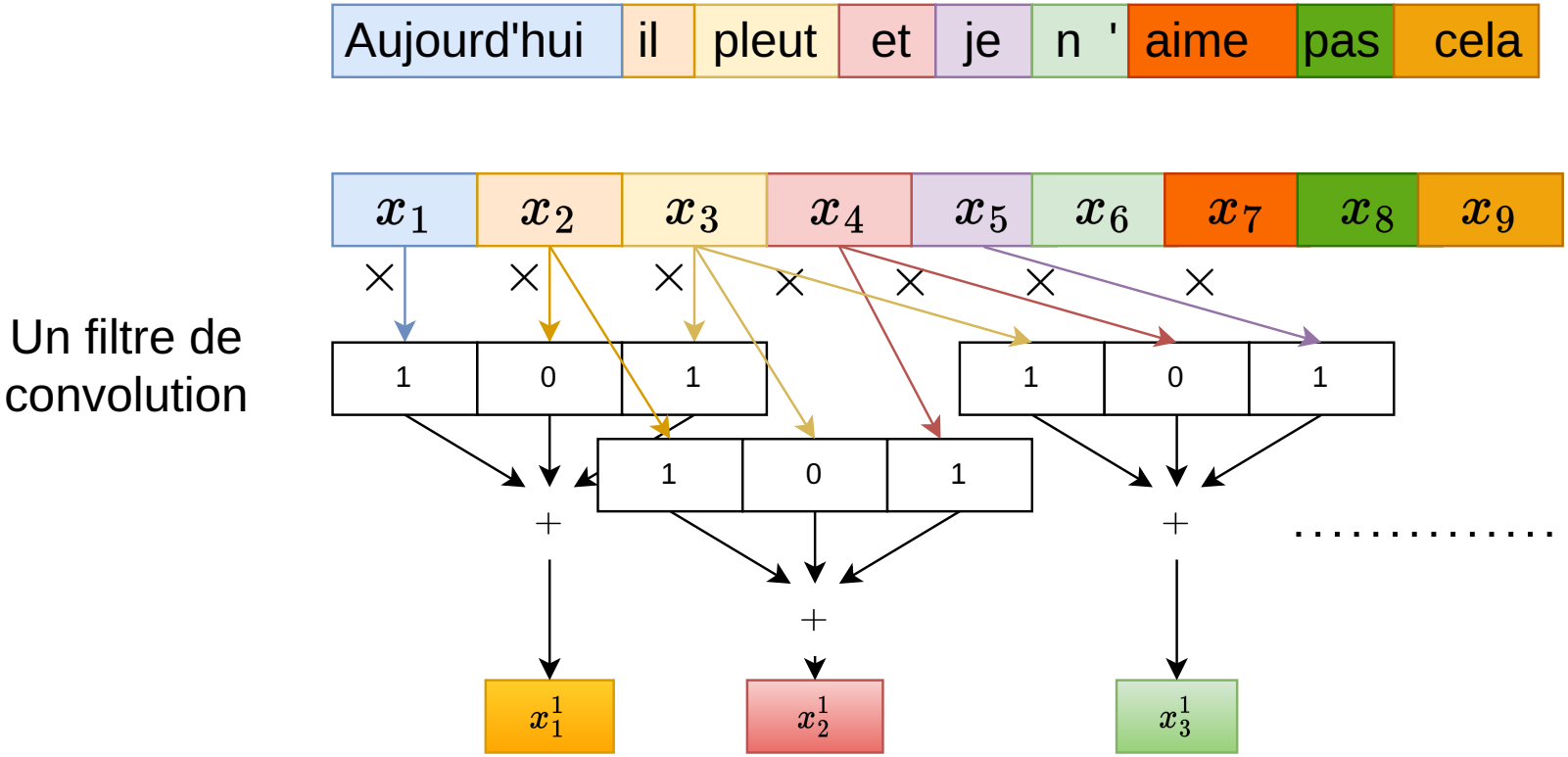
Cependant le MLP n'est plus du tout adapté

- Pour un MLP, chaque dimension a son propre sens
- Pas de possibilités de permuter les dimensions, de définir des permutations

$\Rightarrow$ Impossible de traiter les invariances

$\Rightarrow$ Impossible de traiter des longueurs variables

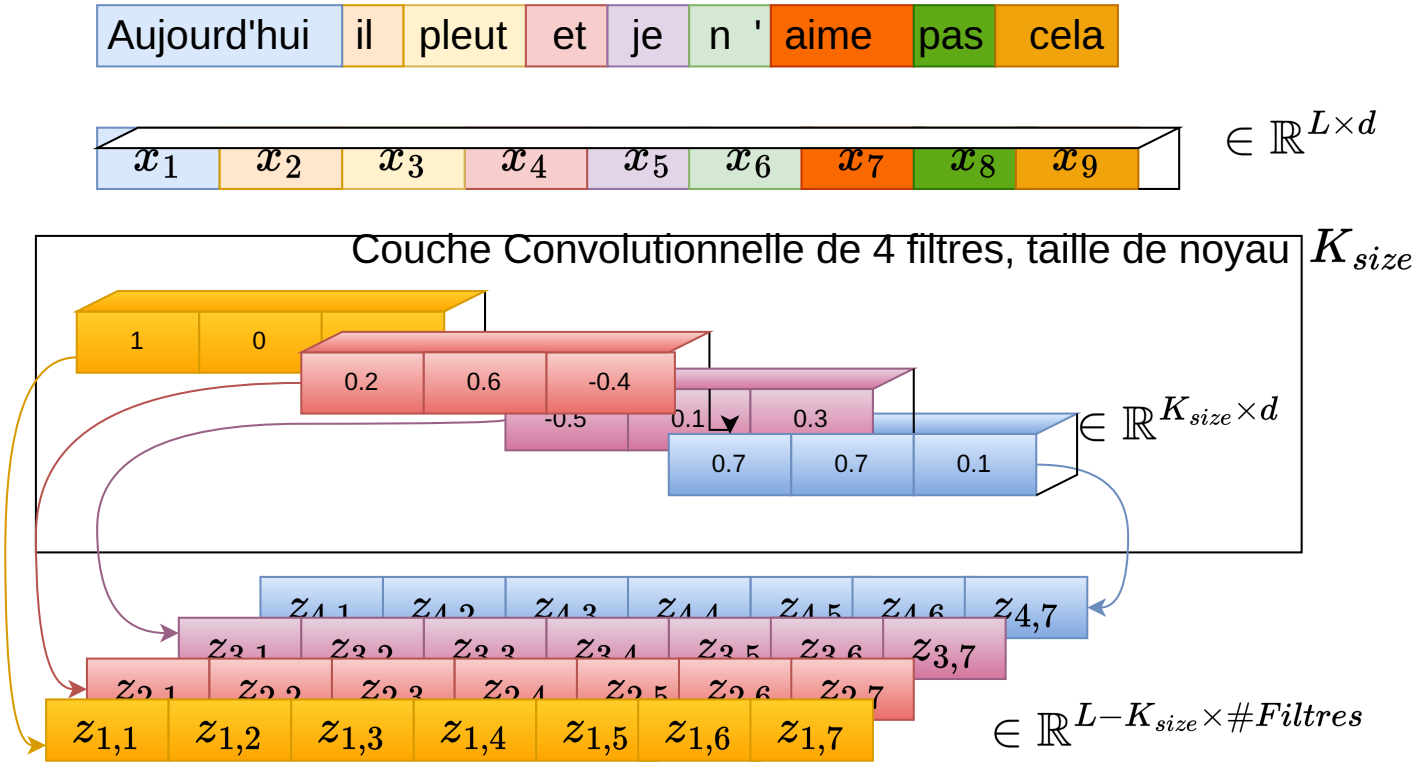
Architecture convolutionnelle 1D



Principe d'une convolution

- Filtre qui effectue une somme pondérée sur une petite fenêtre des entrées
- Les poids du filtre détermine la fonction du filtre
- Le filtre est passé itérativement sur toute la séquence et produit une sortie scalaire à chaque position

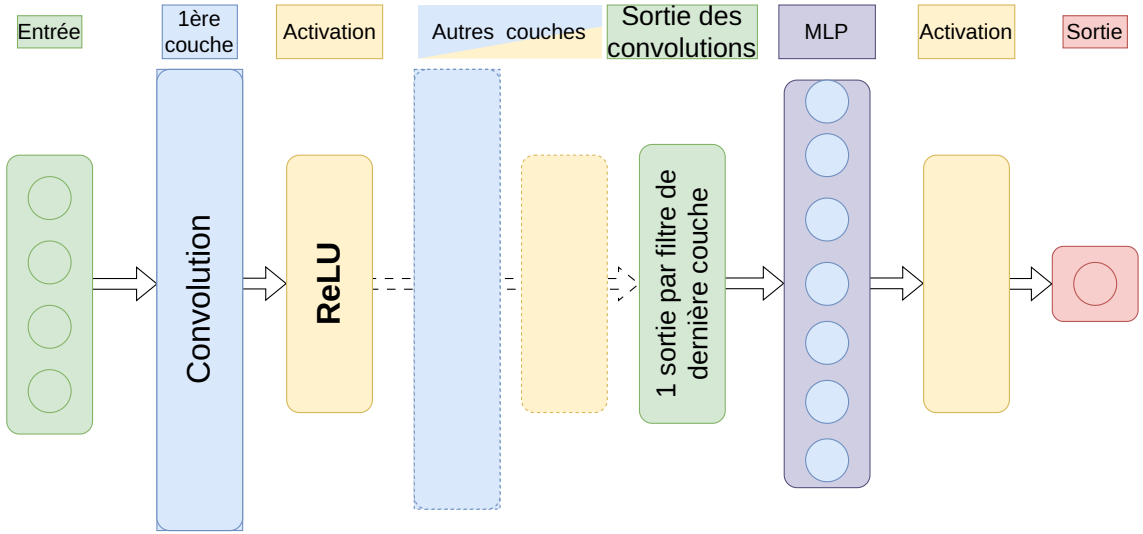
Architecture convolutionnelle 1D



Caractéristiques d'une convolution

- Taille du noyau : taille de la fenêtre glissante
- Stride : déplacement du noyau à chaque itération
- La convolution peut être définie en 1 dimension, en 2, . . . tout dépend du type de signal en entrée (texte, image, vidéo, séries temporelles)

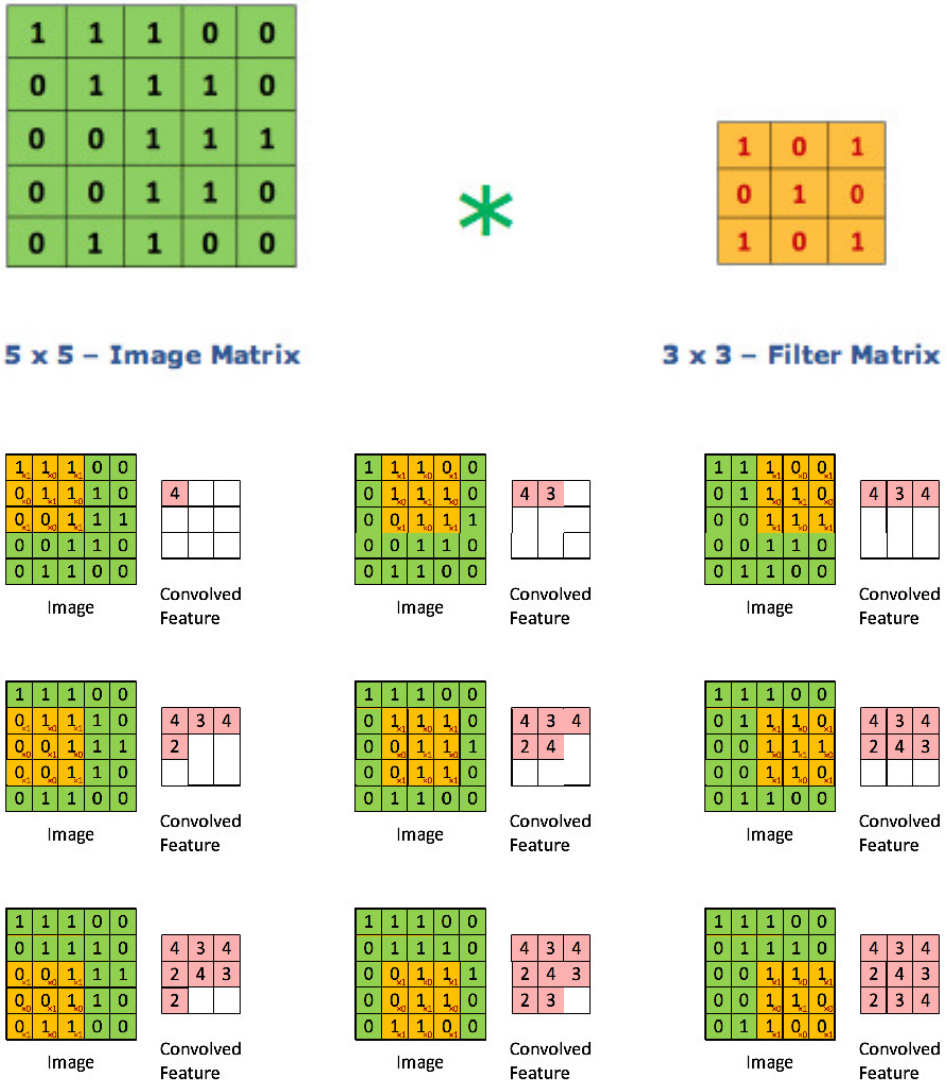
Architecture convolutionnelle 1D



Utilisation pour de la classification

- Série de convolutions qui permettent de repérer la présence ou l'absence de certaines caractéristiques linguistiques
- Pour obtenir un vecteur de dimension fixe, on peut prendre le maximum de chaque filtre de la dernière couche (max-pooling global).
- La sortie des couches convolutionnelles est donc un vecteur de taille le nombre de filtres de la dernière couche, chaque filtre spécialisé pour une caractéristique particulière
- Utilisation d'un MLP pour la tâche subséquente.

Convolution 2D



Application d'une transformation linéaire sur toutes les régions de l'image

Couche de *Pooling*

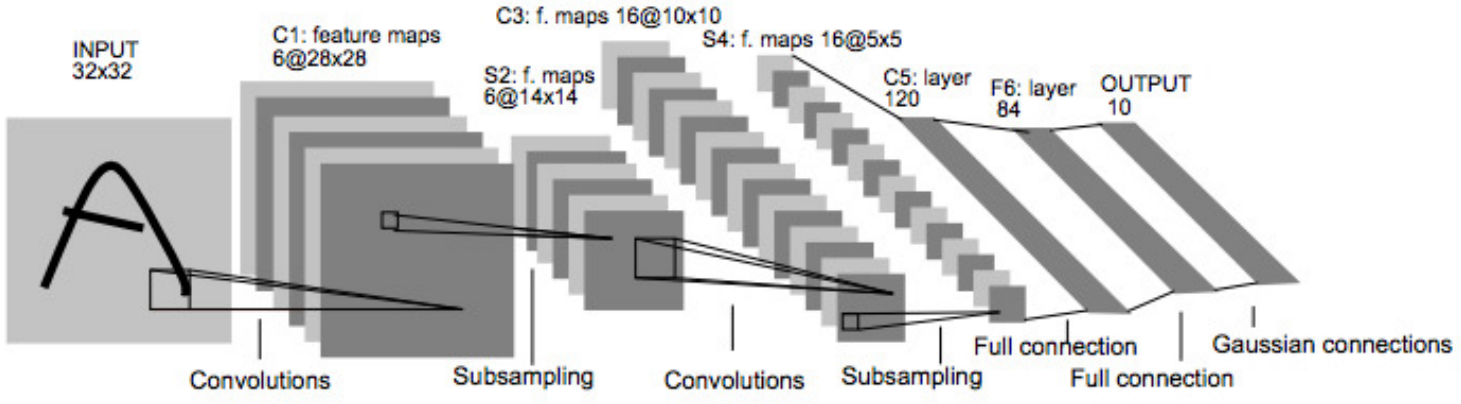


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Pooling (ou subsampling) : Réduire la dimensionalité de sortie

- Max Pooling : on prend le max sur une fenêtre
- Average Pooling : on fait la moyenne
- Sum Pooling : la somme

...

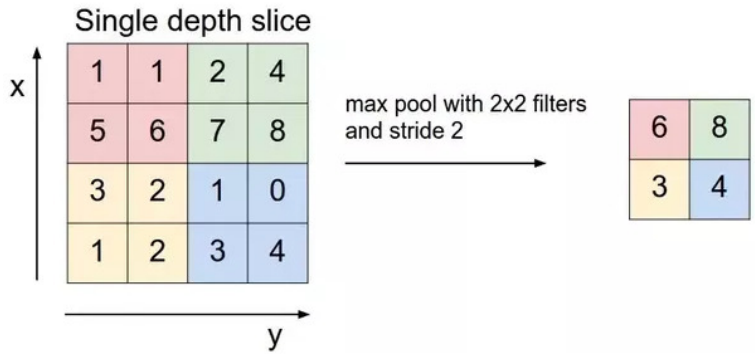
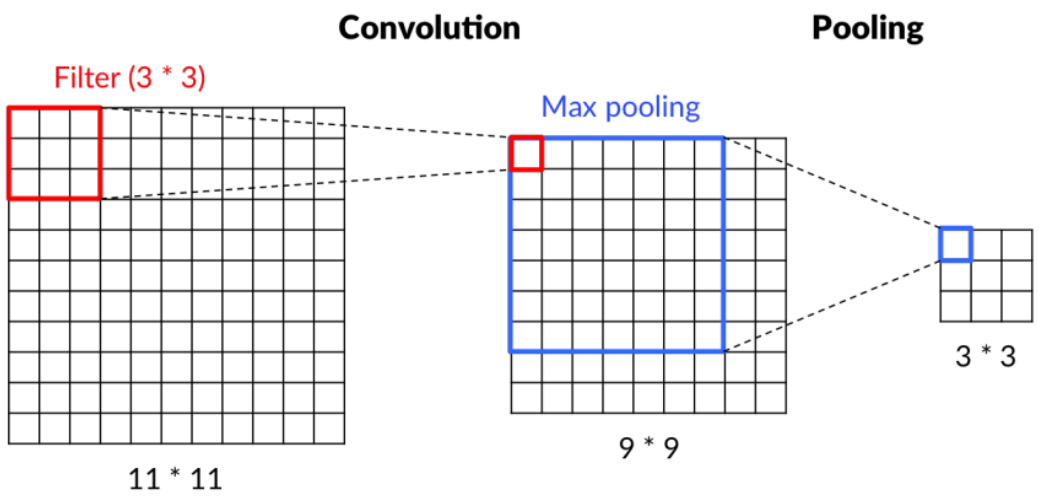
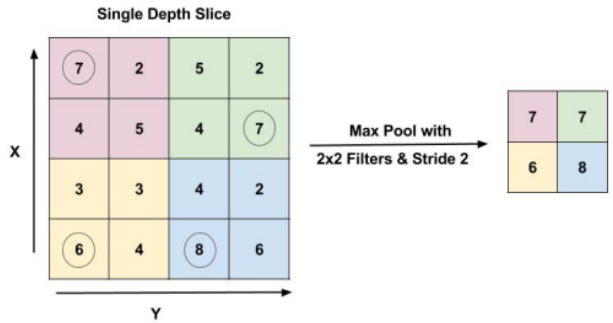
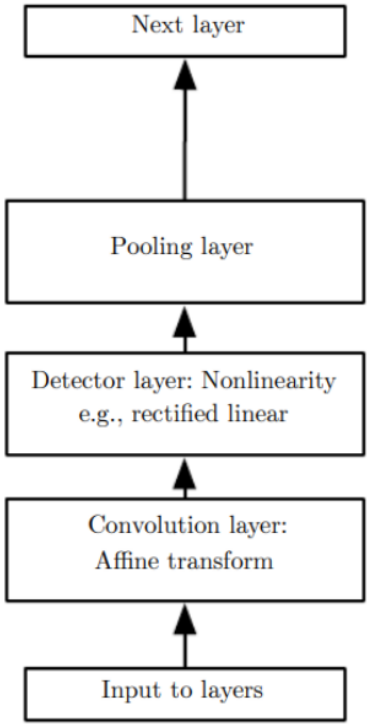


illustration :
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

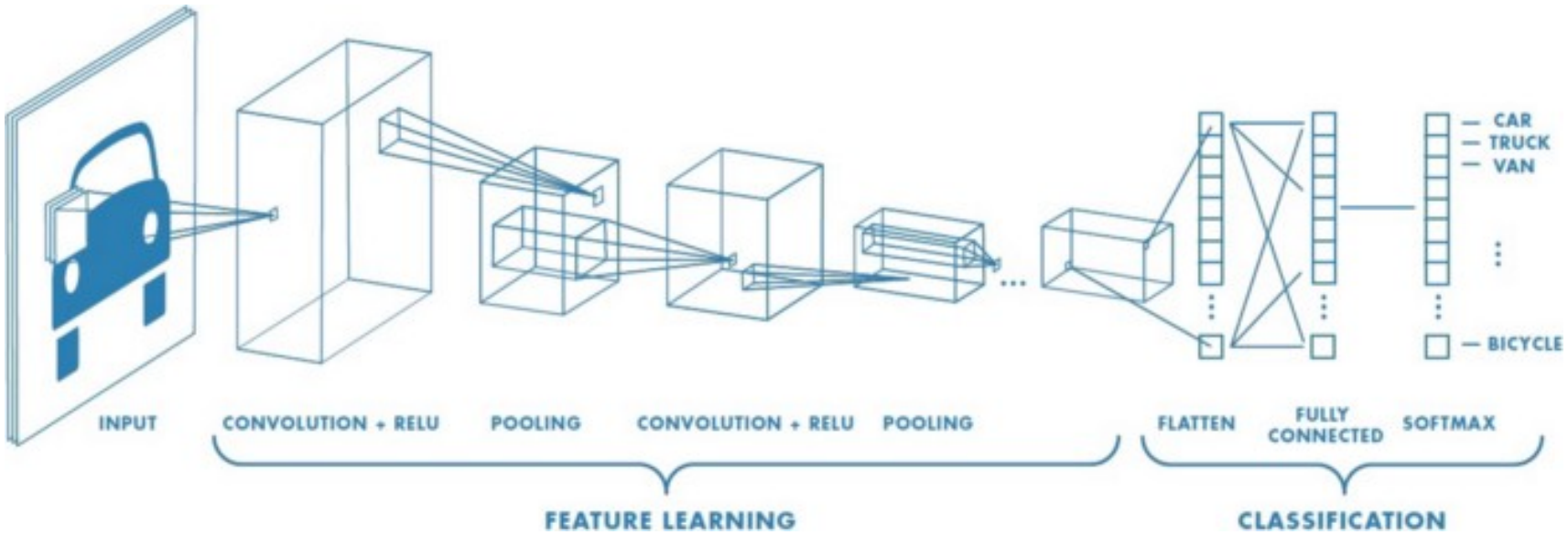
Couche convolutionnelle usuelle





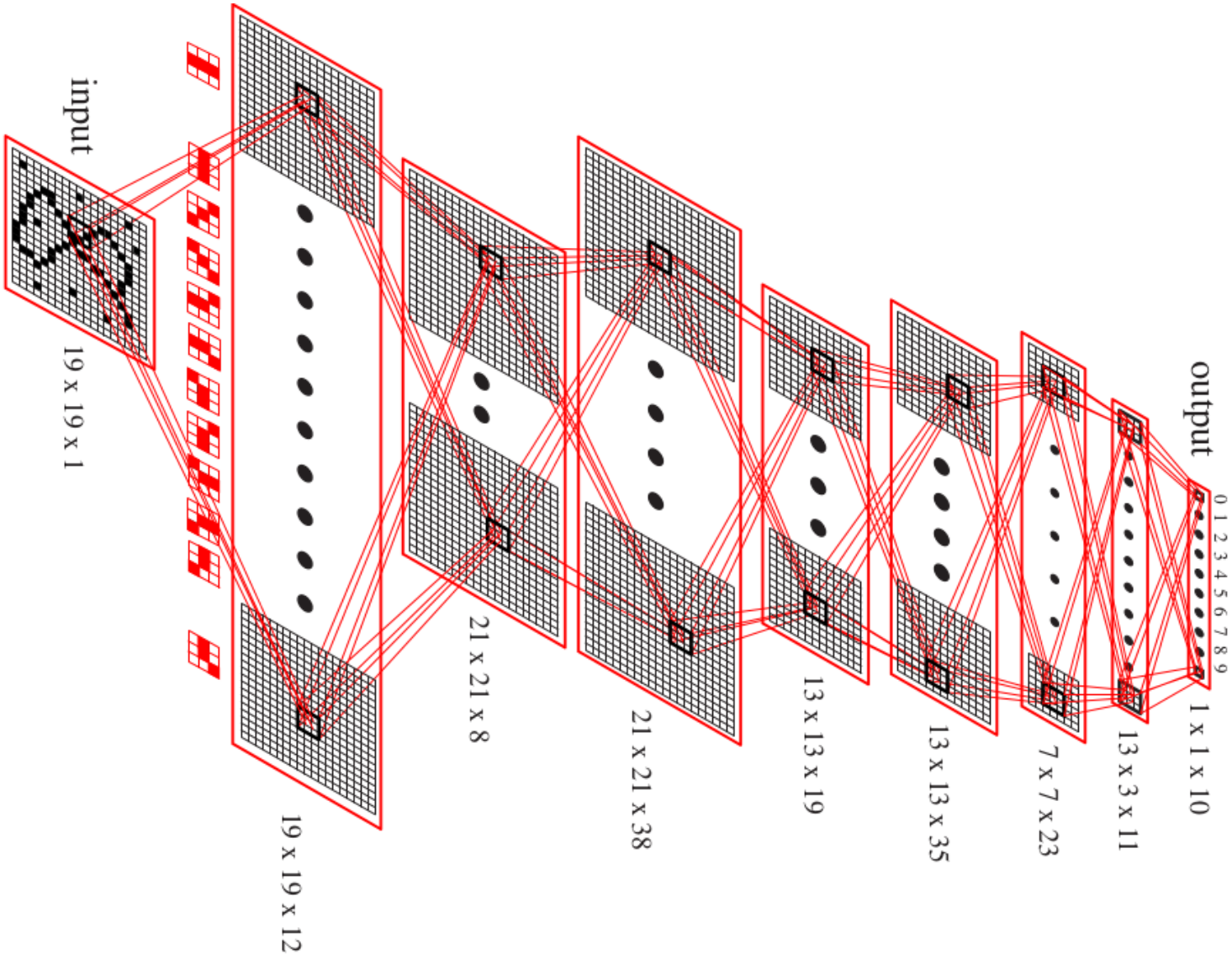


Réseaux convolutifs



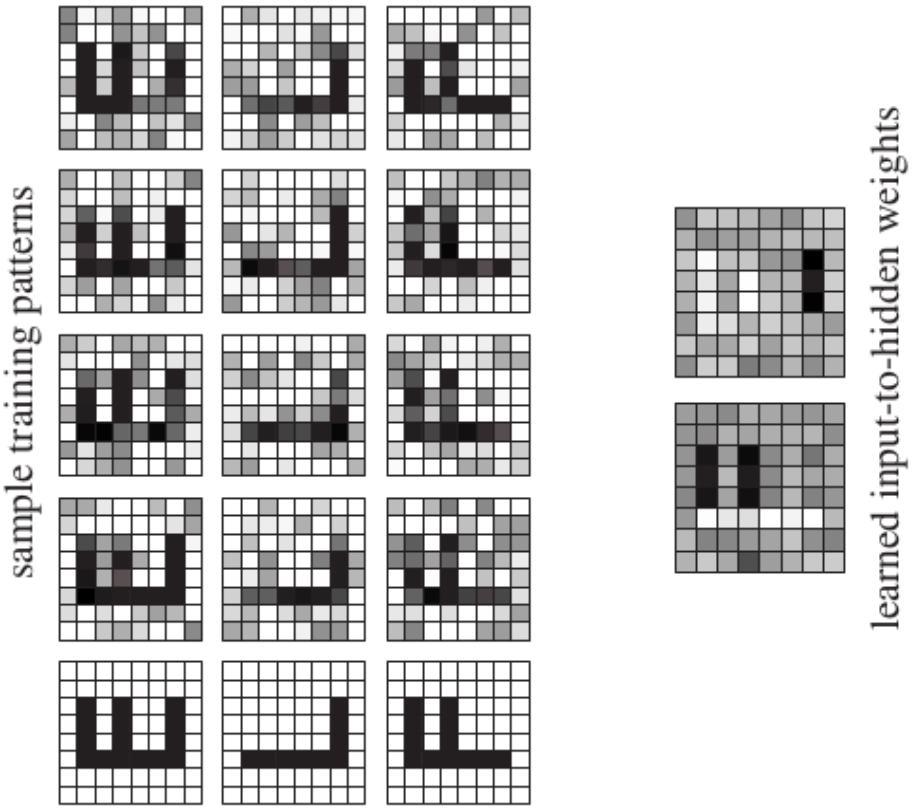
Exemple

Reconnaissance de caractères [Duda et al 00]

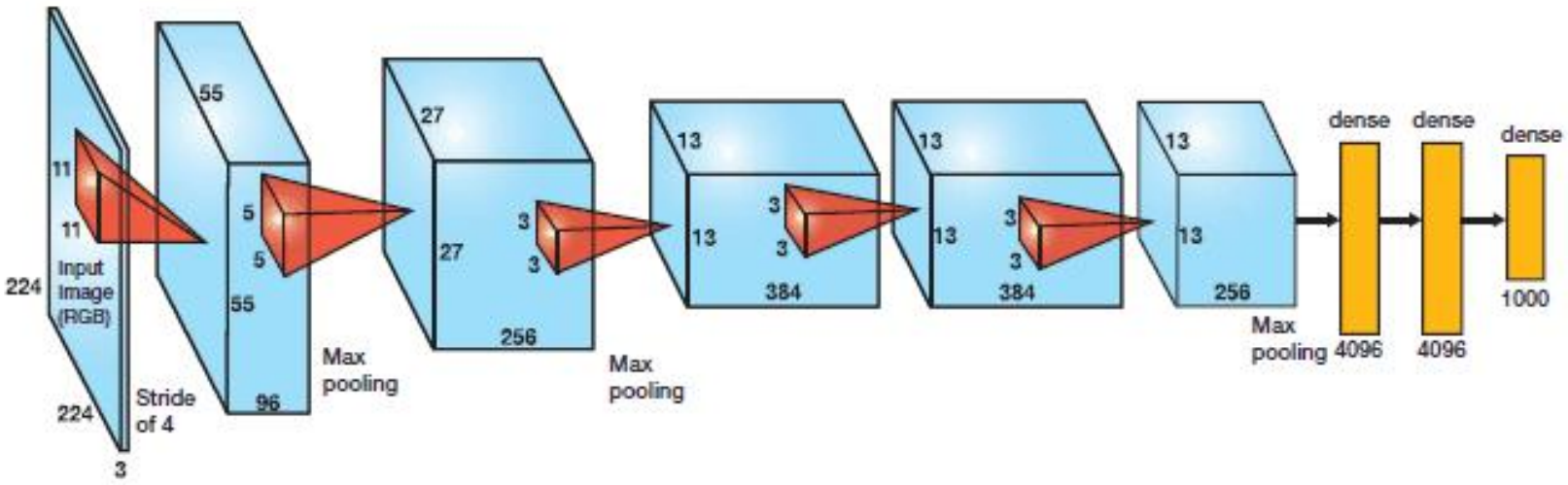


Exemple

Reconnaissance de caractères (couches internes) [Duda et al 00]

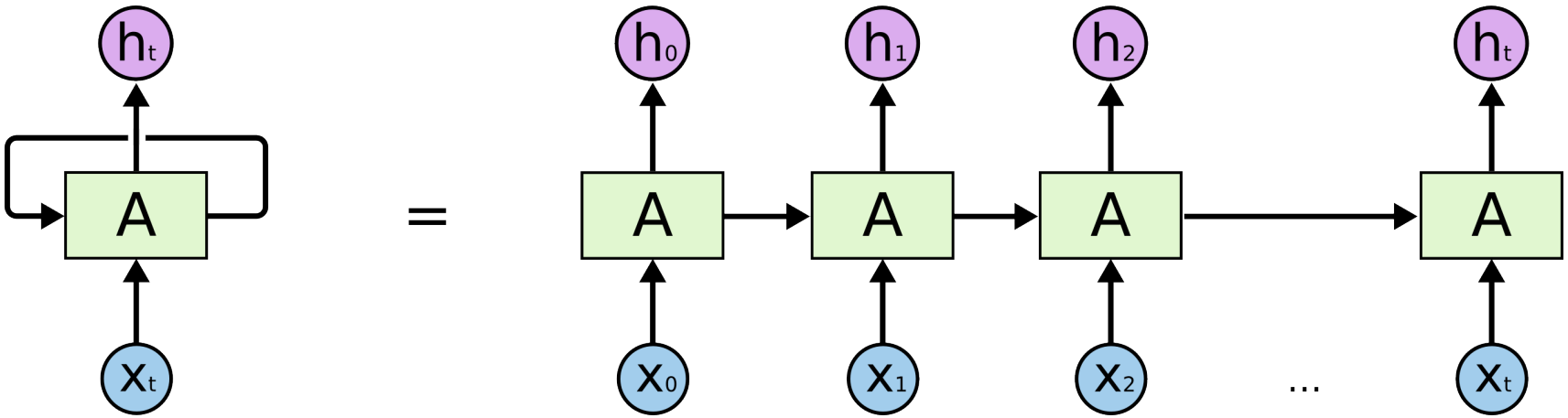


AlexNet (2012)



- $11 \times 11, 5 \times 5, 3 \times 3$ convolutions
- Max-pooling, ReLU activations
- Dropout et Data-augmentation

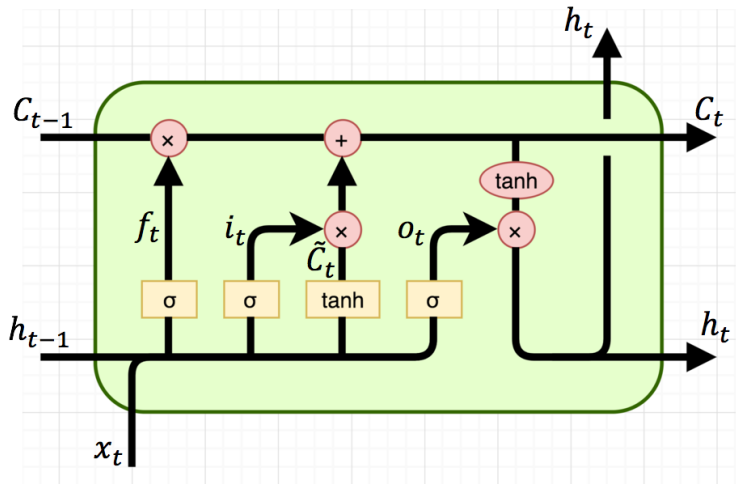
Réseaux récurrents



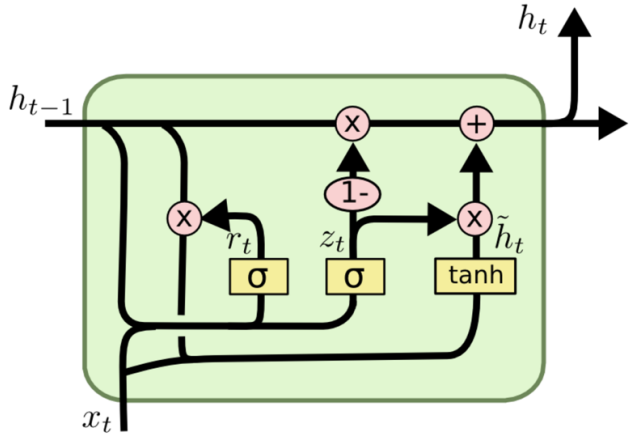
Pour les séquences

- L'objectif est de prédire l'élément suivant d'une séquence : $(x^0, x^1, \ldots, x^t)$
- Hypothèse : $x^{t+1} = f(x^t, h^t)$ l'élément suivant d'une séquence dépend des éléments précédents et d'un état mémoire latent.
- Le réseau prédit pour chaque passage l'état h^t latent.
- Un autre réseau est utilisé pour "décoder" l'état mémoire en la valeur x^t associée.

Long Short Term Memory (LSTM) et Gated Recurrent Unit (GRU)



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

Deux variantes des RNNs

- Un processus de mémoire mis-à-jour automatiquement
- Permet de se ”souvenir” (d’avoir une information persisente d’état en état)

Et beaucoup d'autres

- RBMs (Restricted Boltzmann Machine)
- V.A.E. (Variational Auto-encoder)
- G.A.N. (Generative Adversial Network)
- Mécanisme d'attention (globale et self-attention)
- Stochastic Unit (Reinforce, Gumbel-Softmax, Straight-Through estimator)

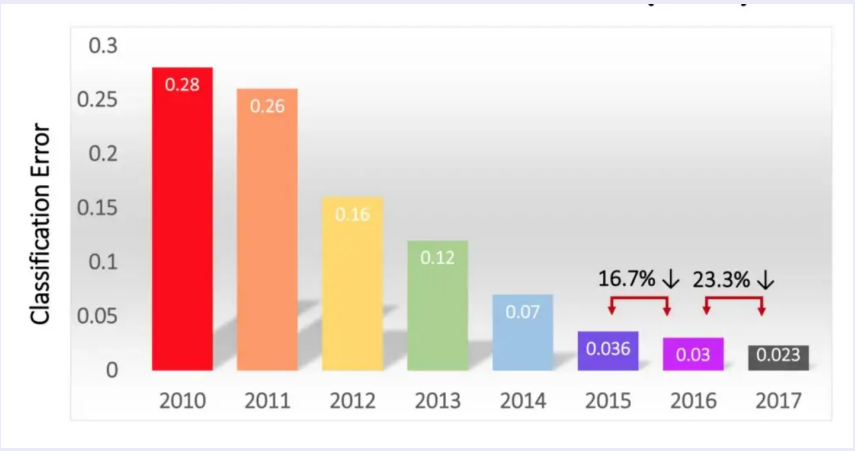
Plan

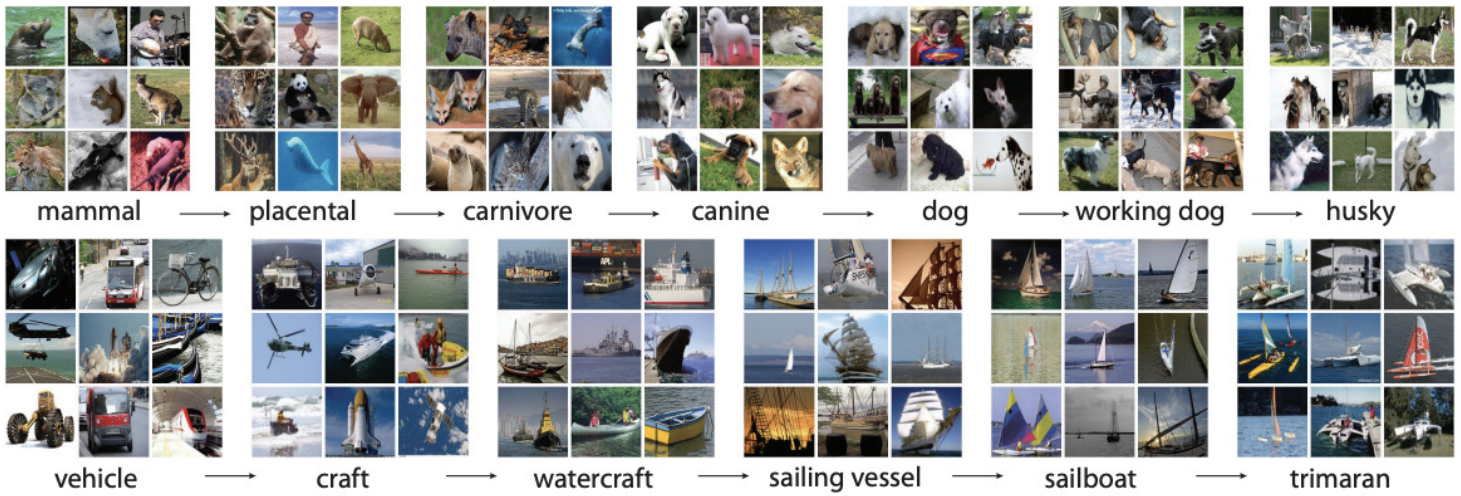
Quelques révolutions du Deep Learning - 1

ImageNet

- Challenge de classification d'images 2009
- Arrivée du Deep en 2012 - écrase la concurrence
- Emergences des modèles les plus connus : AlexNet, Inception, ResNet, ViT...

Résultats depuis 2010



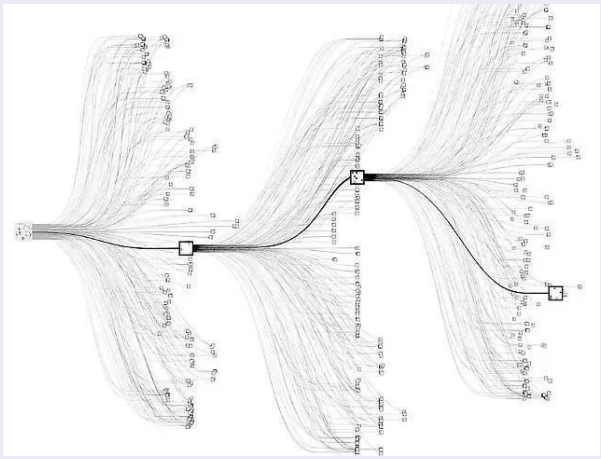


Quelques révolutions du Deep Learning - 2

Deep Q-Learning/Alpha[Go/Zero]

- Capable de jouer à des jeux génériques sans connaître les règles
- 2015 : bat pour la première fois un joueur professionnel de Go
- 2017 : bat le champion du monde de Go

Arbre d'exploration

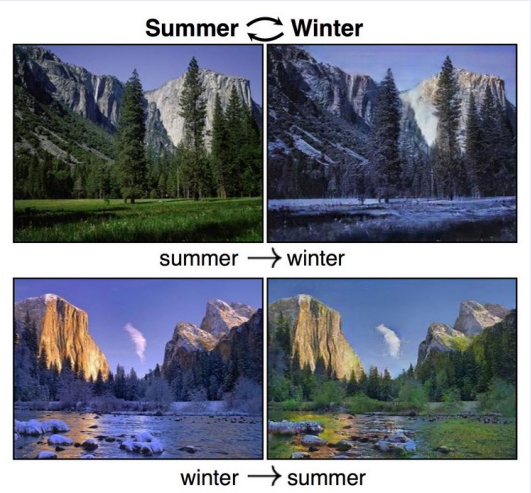


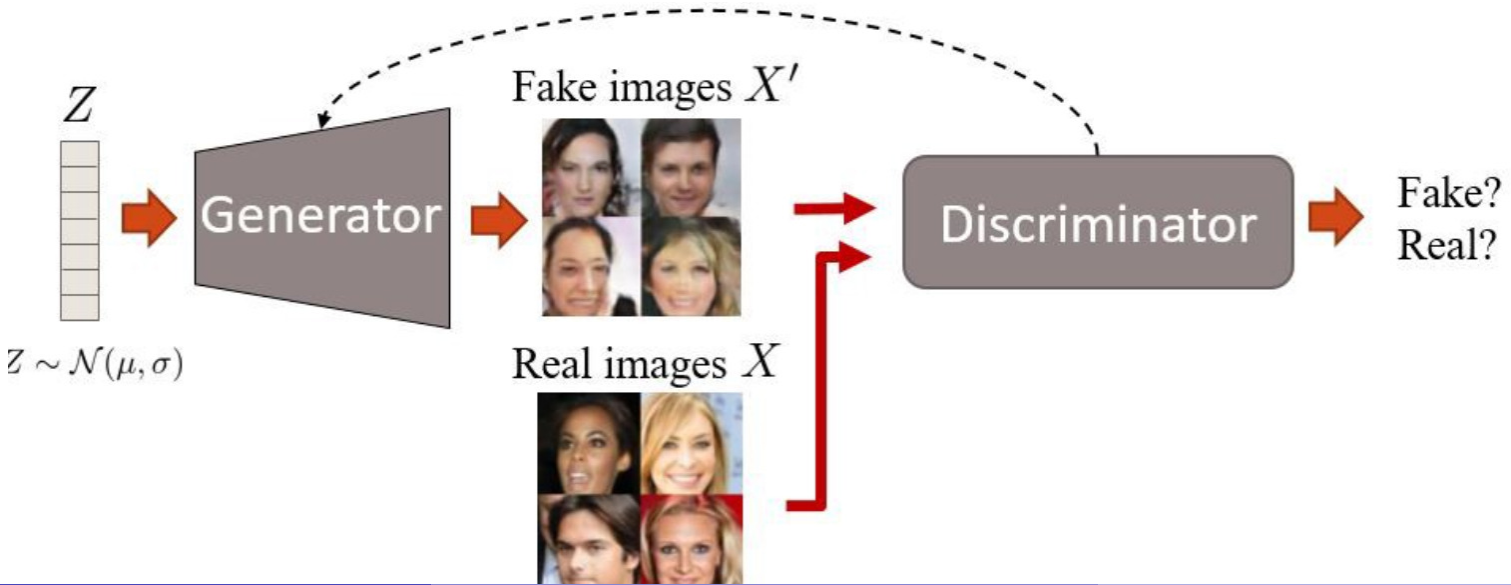
Quelques révolutions du Deep Learning - 3

Generative Adversarial Network (2014)

- Nouveau paradigme d'apprentissage : faussaire contre policier
- Enormément de domaines d'application, en particulier la génération d'image
- Capable d'apprendre des caractéristiques latentes

Transfert de style





Quelques révolutions du Deep Learning - 4

Modèles de Langues

- Origine : word2vec (2013) (apprentissage de représentation), explosion des performances avec le concept d'attention (Transformer, 2017)
- Ont révolutionné tout en TAL : traduction, ChatBot, Recherche d'Information, ...
- Multiples modèles : GPT, BERT, ELMO, Galactica

ChatGPT chatbot

CB What is brandolini's law?

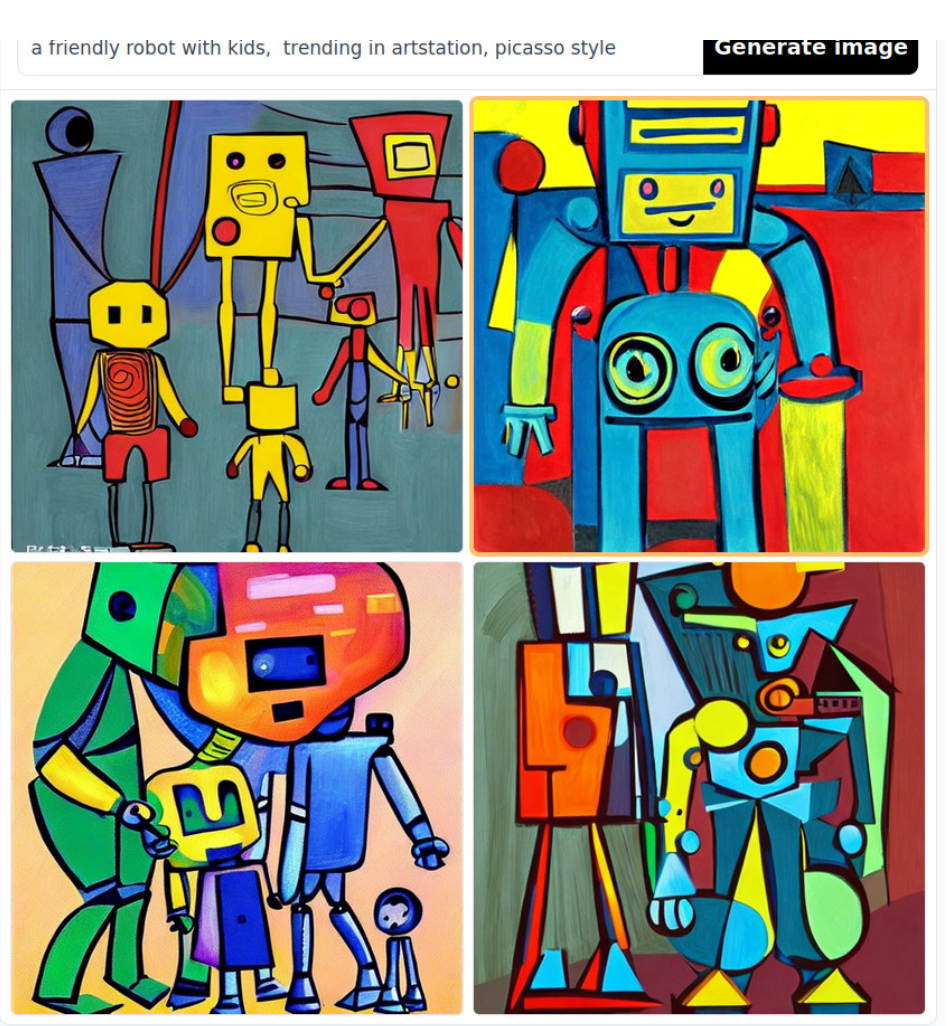
Brandolini's law, also known as the "Bullshit Asymmetry Principle," is a humorous aphorism that states: "The amount of energy needed to refute bullshit is an order of magnitude bigger than to produce it." In other words, it takes more effort and time to debunk a false or misleading statement than it does to make the statement in the first place.

The law is named after Alberto Brandolini, an Italian software developer who coined the term in a tweet in 2013. Brandolini's law is often used to describe the challenges of debunking misinformation or false claims, particularly in the age of social media, where false and misleading information can spread quickly and easily.

While Brandolini's law is not a formal scientific principle, it is a useful way of illustrating the challenges of combating misinformation and the importance of carefully evaluating information before accepting it as true.

Et quand on mélange toutes ces révolutions . . .

On obtient *DALL-E*, *Stable Diffusion* et autres modèles du genre



La suite en M2 ! (et en projet . . .)