# R6.DWeb-DI.02
# Introduction au Machine Learning avec ML5

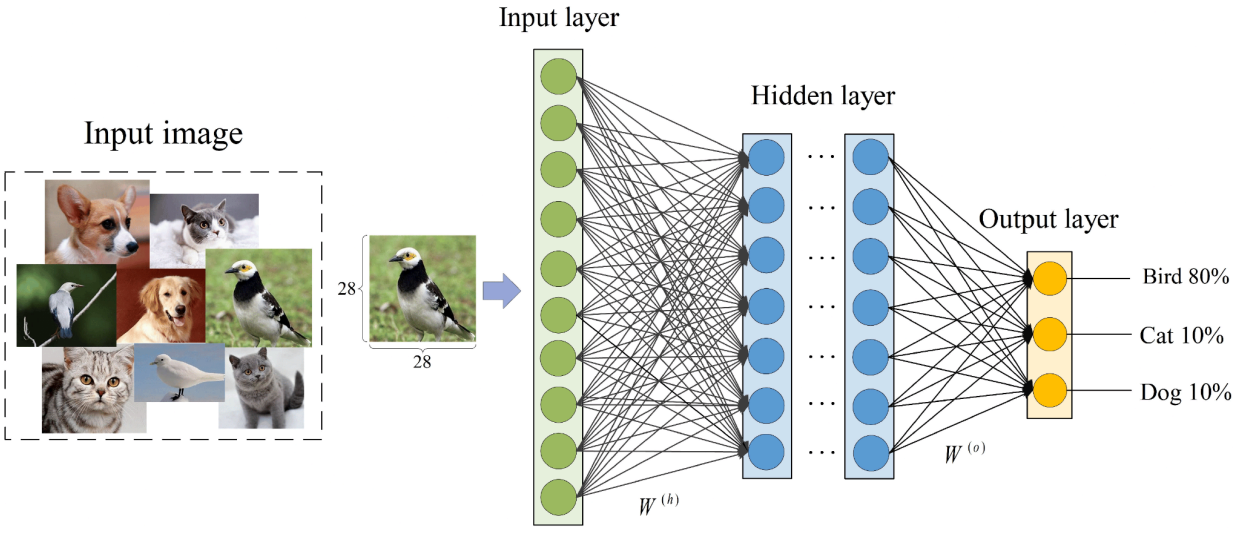
---

**Calendrier**

14 Janv. 2025, Mar.10:00 à 12:00 --- Exo 1
17 Janv. 2025, Ven. 09:00 à 12:00 --- Exos 2 & 3
21 Janv. 2025, Mar. 10:00 à 12:00 --- Exo 4
24 Janv. 2025, Ven. 09:00 à 12:00 --- Exo 5
27 Janv. 2025, Lun. 15:30 à 17:30 --- Exo 5
29 Janv. 2025, Mer. 14:00 à 17:00 --- Eval

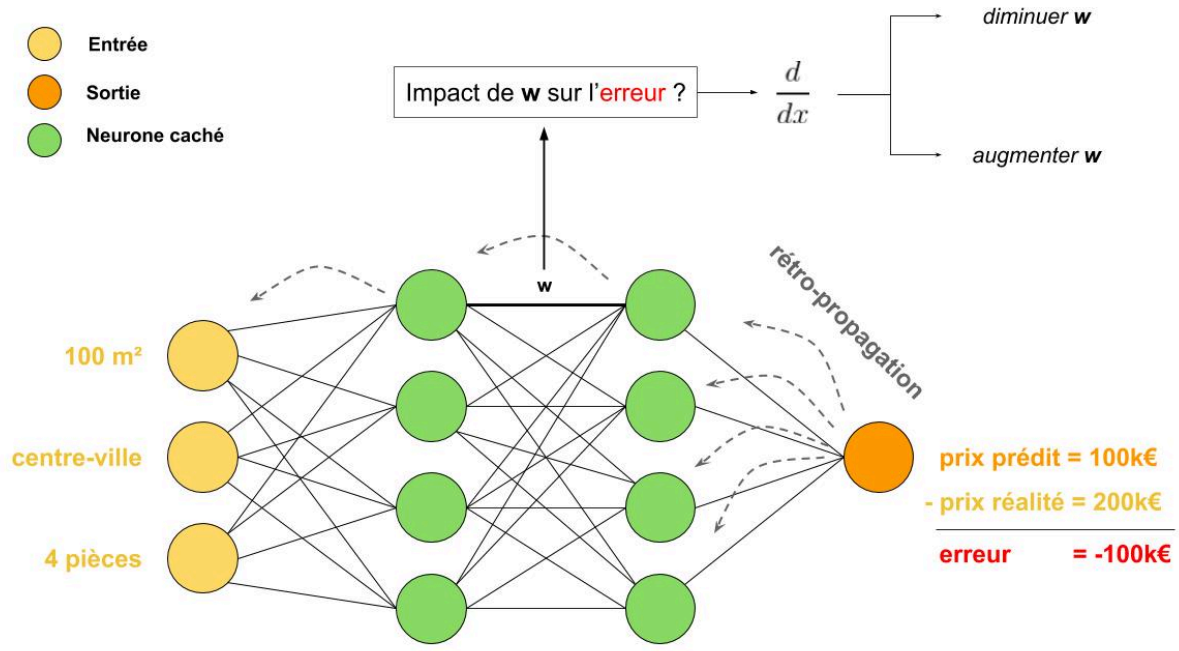
---

La librairie ML5 donne accès à différents modèles de “**réseaux de neurones”**, certains déjà entraînés sur des bases de données contenant plusieurs milliers d’images. Le premier concept que nous allons voir est celui de la **classification**, qui permet, à partir de données d’entrée, de produire en sortie un vecteur de probabilités qui indique, pour chaque classe, si ces données correspondent.




Dans le schéma ci-dessus les données d’entrée sont des images, mais en réalité n’importe quelle entrée convertible en nombre peut convenir. Commencez par tester en ligne les exemples simples permettant de déterminer si une couleur est plutôt rouge, verte ou bleue, ou bien d’estimer dans quelle direction va un mouvement de souris :
https://docs.ml5js.org/#/reference/neural-network

On voit ici au début du programme la première étape de tout algorithme d'IA qui consiste à entraîner le réseau, c'est-à-dire déterminer quels poids donner aux différentes combinaisons de "neurones". C'est ce qu'on appelle la phase de "rétro-propagation", expliquée par le schéma ci-dessous pour un réseau qui essaie de prédire le prix de vente d'une maison en fonction de différents paramètres :




On trouve aussi dans le code les concepts de "epochs" et de normalisation qui sont essentiels pour obtenir des résultats (à peu près) corrects.

**Exo 1 : Classification vs Régression**

1. Trouvez comment utiliser p5js et ml5 en local (à partir de https://unpkg.com/browse/ml5@1.2.1/dist/ml5.min.js). Testez ensuite en local les exemples précédents, puis à partir des données ci-dessous classifiez un animal de taille = 63, poids = 22 et age = 3 :

```
const data = [
  { taille: 82, poids: 14, age: 10, animal: "chat" },
  { taille: 80, poids: 23, age: 10, animal: "chien" },
  { taille: 43, poids: 24, age: 13, animal: "chien" },
  { taille: 45, poids: 39, age: 10, animal: "chien" },
  { taille: 53, poids: 20, age: 10, animal: "chien" },
  { taille: 54, poids: 23, age: 6, animal: "chat" },
  { taille: 37, poids: 15, age: 1, animal: "chat" },
  { taille: 59, poids: 45, age: 12, animal: "chien" },
  { taille: 61, poids: 25, age: 5, animal: "chat" },
```

```
  { taille: 68, poids: 35, age: 13, animal: "chien" },
  { taille: 49, poids: 22, age: 4, animal: "chat" },
  { taille: 70, poids: 41, age: 8, animal: "chien" },
  { taille: 42, poids: 18, age: 3, animal: "chat" },
  { taille: 77, poids: 50, age: 7, animal: "chien" },
  { taille: 56, poids: 26, age: 6, animal: "chat" },
  { taille: 64, poids: 33, age: 9, animal: "chien" },
  { taille: 38, poids: 16, age: 2, animal: "chat" },
  { taille: 72, poids: 47, age: 11, animal: "chien" },
  { taille: 55, poids: 21, age: 4, animal: "chat" },
  { taille: 67, poids: 38, age: 8, animal: "chien" },
  { taille: 47, poids: 20, age: 3, animal: "chat" },
  { taille: 78, poids: 48, age: 9, animal: "chien" },
  { taille: 50, poids: 22, age: 5, animal: "chat" },
  { taille: 65, poids: 36, age: 7, animal: "chien" },
  { taille: 39, poids: 17, age: 2, animal: "chat" },
  { taille: 74, poids: 44, age: 10, animal: "chien" },
  { taille: 58, poids: 25, age: 6, animal: "chat" },
  { taille: 69, poids: 40, age: 8, animal: "chien" },
  { taille: 41, poids: 19, age: 3, animal: "chat" },
  { taille: 75, poids: 46, age: 9, animal: "chien" },
  { taille: 52, poids: 23, age: 5, animal: "chat" },
  { taille: 66, poids: 37, age: 7, animal: "chien" },
  { taille: 40, poids: 18, age: 2, animal: "chat" },
  { taille: 71, poids: 43, age: 11, animal: "chien" },
  { taille: 57, poids: 24, age: 6, animal: "chat" },
  { taille: 63, poids: 32, age: 8, animal: "chien" },
  { taille: 44, poids: 20, age: 3, animal: "chat" },
  { taille: 76, poids: 49, age: 9, animal: "chien" },
  { taille: 51, poids: 22, age: 5, animal: "chat" },
  { taille: 62, poids: 30, age: 7, animal: "chien" },
  { taille: 48, poids: 21, age: 4, animal: "chat" },
  { taille: 73, poids: 45, age: 10, animal: "chien" },
  { taille: 60, poids: 26, age: 6, animal: "chat" },
  { taille: 68, poids: 34, age: 8, animal: "chien" },
  { taille: 46, poids: 19, age: 3, animal: "chat" },
  { taille: 79, poids: 50, age: 9, animal: "chien" },
  { taille: 53, poids: 23, age: 5, animal: "chat" },
  { taille: 64, poids: 31, age: 7, animal: "chien" },
  { taille: 42, poids: 18, age: 2, animal: "chat" },
  { taille: 70, poids: 42, age: 8, animal: "chien" }
];
```

2. Comme on l'a vu, un réseau de neurones produit en sortie des valeurs qu'on peut interpréter comme des probabilités. En pratique on peut aussi considérer que ces valeurs sont des estimations du résultat recherché, on parle alors de **régression**. Le fonctionnement est identique, mais certains paramètres sont différents:
    - le paramètre 'task' doit être fixé à 'regression' au lieu de 'classification'
    - l'opération à appliquer n'est plus 'classify' mais 'predict'

- appliquez ce principe pour estimer le poids d'un chien de taille = 67 et age = 12

**Exo 2 : Classification d'images avec des modèles pré-entraînés**

La librairie fournit différents modèles déjà entraînés, permettant de détecter quel(s) objet(s) apparaît sur une image ou une vidéo. Reprenez l'exemple de base donné sur https://docs.ml5js.org/#/reference/image-classifier

1. Testez l'exemple de base qui reconnaît un oiseau avec les modèles "mobilenet", "darknet" et "darknet-tiny", testez aussi avec d'autres images et comparez les résultats :
   - File:Travaille sur mouton.jpg - Wikimedia Commons
   - File:Vache-de-race-limousine-en-correze-2.jpg - Wikimedia Commons
   - https://cdn.futura-sciences.com/buildsv6/images/square1280/2/b/7/2b7016e931_50212534_yeti.jpg
2. Testez avec webcam/vidéo, puis essayez de lister les limites de cette approche
   - https://www.pexels.com/search/videos/animal/
3. Testez le modèle "DoodleNet" en local à partir de l'exemple fourni

**Exo 3 : Entraînement et classification sur vos propres images**

Il est possible d'entraîner un modèle avec vos propres images :
https://docs.ml5js.org/#/reference/image-classifier-tm

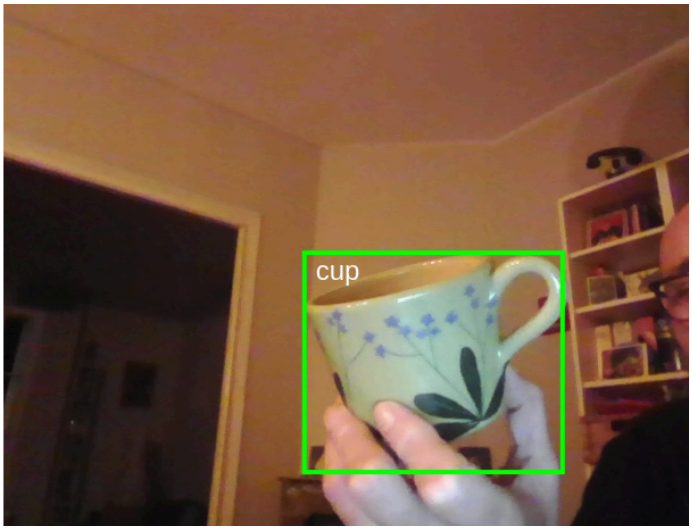
Suivez les étapes pour classifier automatiquement des éléments visibles avec la webcam :

1. Commencez avec deux classes d'objets, entraînez le modèle sur Teachable Machine avec une cinquantaine d'images par classe, en utilisant la webcam et en variant l'orientation, le fond, la distance à la caméra, etc; testez en ligne puis en local sur votre PC et vérifiez si vos objets sont reconnus avec une bonne probabilité
2. Essayez ensuite de reconnaître les gestes "pouce vers le haut" ou "pouce vers le bas" en entraînant le modèle avec différentes personnes, puis testez-le avec une nouvelle personne
3. Et si on jouait à Pierre-Feuille-Ciseaux ?

**Exo 4 : Détection de la position des objets**

L'ancienne version de la librairie comporte quelques modèles qui n'apparaissent pas encore dans la version 2024. Ils fonctionnent toujours néanmoins (attention à bien utiliser l'ancienne version de la lib) et permettent de détecter des objets avec un modèle pré-entraîné, ou entraîné sur vos propres objets.

1. Reprenez les exemples pour détecter des objets avec la webcam à partir de cette page : https://archive-docs.ml5js.org/#/reference/object-detector



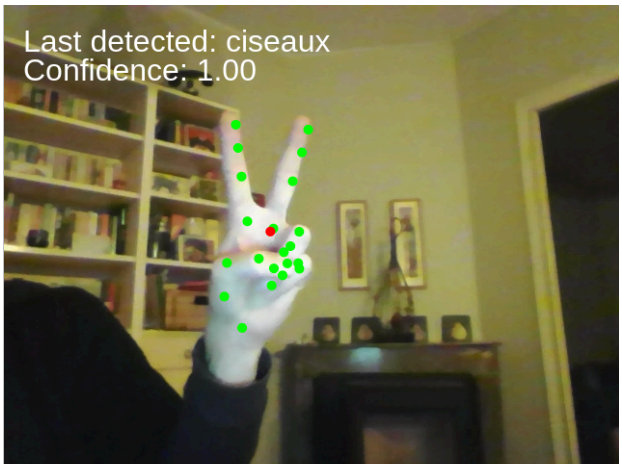

2. Une autre stratégie consiste à ré-utiliser un réseau de neurones entraîné pour la reconnaissance d'images comme avec Teachable Machine, mais en l'entraînant à prédire une valeur continue (c'est donc une régression, comme dans l'exercice 1.2 mais avec des images comme données d'entrée) :
   https://archive-docs.ml5js.org/#/reference/feature-extractor
   Regardez les exemples puis utilisez ce principe pour coder une appli capable de déterminer la position X d'un objet :
   - le code débute par la phase d'entraînement : à l'aide d'un slider et d'un bouton, laissez l'utilisateur capturer des images en positionnant l'objet plutôt à gauche, au centre ou à droite
   - une fois qu'il a capturé suffisamment d'images il lance l'entraînement
   - ensuite l'appli devrait afficher la position approximative de l'objet sans trop d'erreurs…
   - sur le même principe essayez de détecter aussi la position Y

**Exo 5 : Détection du visage, des mains ou du squelette**

La dernière version de la librairie propose des modèles pré-entraînés qui marchent très bien pour détecter les humains derrière la webcam : **BodyPose** pour le squelette, **BodySegmentation** pour séparer la personne visible du fond, **HandPose** pour la position des doigts, **FaceMesh** pour le visage. Regardez et testez les exemples sur ML5.js

1. Utilisez votre corps pour lancer des balles grâce à BodyPose en repérant le poignet et le coude droits (inspiré par
   Combining ml5 PoseNet and p5play to manipulate objects with my body - p5js ) :

2. Par défaut BodyPose utilise un modèle appelé “MoveNet”, que vous pouvez remplacer par un autre modèle plus précis appelé “BlazePose” qui donne accès à la coordonnée Z des points-clefs relative au centre du corps (voir aussi 3D Pose Estimation with ml5.js )
   Ainsi on peut facilement estimer par exemple si un utilisateur plie le bras droit : si on divise la distance entre l’épaule et le poignet par la somme des distances entre l’épaule et le coude et entre le coude et le poignet, on obtient une valeur proche de 0 si le bras est plié, ou 1 si le bras est étendu dans n’importe quelle direction (en pratique le ratio va plutôt de 0.65 à 1). Utilisez ce principe pour créer un Pong contrôlable avec les bras.
3. Combinez HandPose avec un réseau de neurones simples (Exo 1) pour estimer si l’utilisateur fait le geste “pierre”, “feuille” ou “ciseaux” ; l’idée est ici est d’utiliser non pas la position absolue des doigts, mais leur position relative par rapport au centre de gravité de la main, c’est-à-dire la moyenne des positions de chaque point-clé.




   Ce principe peut s’adapter à beaucoup de situations où on cherche à classifier des gestes, des expressions faciales, etc.
4. Combinez FaceMesh et Teachable Machine (Exo 3) pour estimer approximativement si un utilisateur est en train de regarder à gauche ou à droite. L’idée est qu’on peut focaliser l’entraînement sur une petite zone carrée autour de l’un des deux yeux, plutôt que toute l’image :
   - pour l’entraînement on place un objet vers le bord gauche ou le bord droit, que l’utilisateur doit regarder.

- on duplique dans une image les pixels d'un petit carré autour d'un oeil (identifié grâce à FaceMesh), et on sauvegarde cette image dans un dossier correspondant au label Gauche ou Droite.
- on répète ces opérations plusieurs fois puis on entraîne un réseau sur Teachable Machine avec toutes nos petites images pour classifier la direction du regard
- en exportant ce réseau vous devriez maintenant être capable dans une autre appli, toujours en utilisant FaceMesh pour extraire un petit carré autour de l'oeil, de déterminer approximativement si l'utilisateur regarde plutôt à gauche ou à droite.

. 

https://mediaserver.unilim.fr/videos/r6dweb-di02-tp-2025/