

Projet UCIA

COPIL du 6 janvier 2025
Partie technique

Jean-Luc.Charles@mailo.com





Partie technique

- Concepts et outils de l'IA
- Études menées
- Mise en oeuvre avec la carte RPi4
- Démonstration...

IA : Aspects historiques... de 1950 à 2000



STORY BY DATA

HISTORY OF NEURAL NETWORKS

1943-2019

Warren McCulloch & Walter Pitts, wrote a paper on how neurons might work; they modeled a simple neural network with electrical circuits.

1943

Nathaniel Rochester from the IBM research laboratories led the first effort to simulate a neural network.

1949

1950s

1956

John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes.

1957

Donald Hebb reinforced the concept of neurons in his book, *The Organization of Behavior*. It pointed out that neural pathways are strengthened each time they are used.

1982

1981

The **Dartmouth Summer Research Project** on Artificial Intelligence provided a boost to both artificial intelligence and neural networks.

1969

1959

Frank Rosenblatt began work on the Perceptron; the oldest neural network still in use today.

1958

John Hopfield presented a paper to the national Academy of Sciences. His approach to create useful devices; he was likeable, articulate, and charismatic.

1982

Progress on neural network research halted due fear, unfulfilled claims, etc.

Marvin Minsky & Seymour Papert proved the Perceptron to be limited in their book, *Perceptrons*.

Bernard Widrow & Marcian Hoff of Stanford developed models they called ADALINE and MADALINE; the first neural network to be applied to a real world problem.

1982

1985

1997

1998

NOW

US-Japan Joint Conference on Cooperative/Competitive Neural Networks; Japan announced their Fifth-Generation effort resulted in US worrying about being left behind and restarted the funding in US.

American Institute of Physics began what has become an annual meeting - **Neural Networks for Computing**.

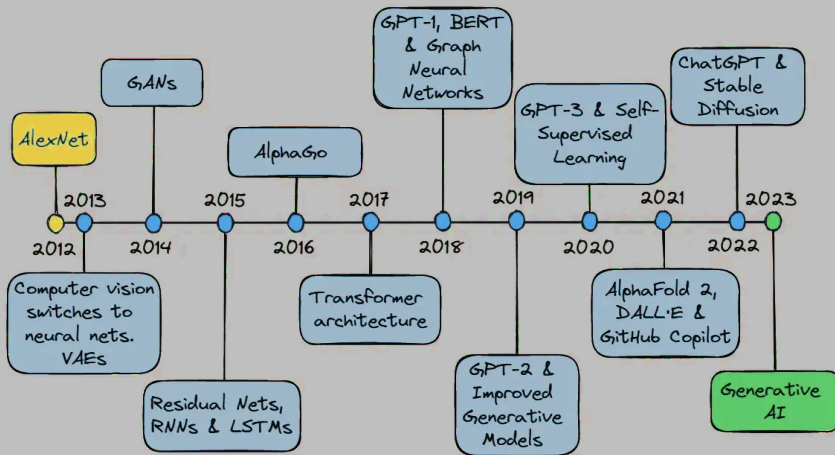
A recurrent neural network framework, LSTM was proposed by **Schmidhuber & Hochreiter**.

Yann LeCun published *Gradient-Based Learning Applied to Document Recognition*.

Neural networks discussions are prevalent; the future is here!

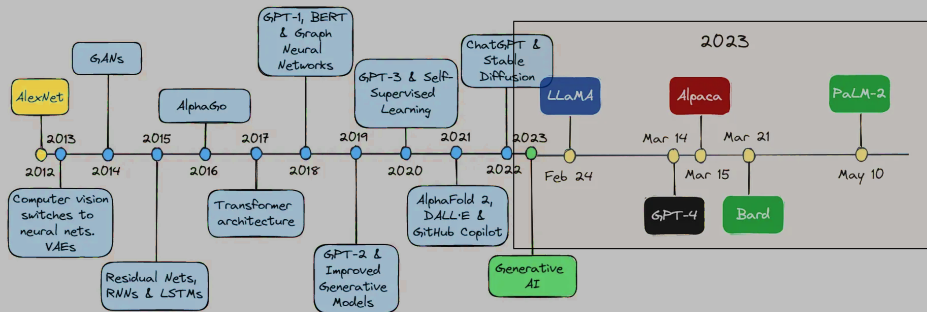
Source: Kate Strachni: "Brief History of Neural Netowrks", medium.com

IA : Aspects historiques... l'accélération post 2012



Source: Thomas A Dorfe: "Ten Years of AI in Review", [medium.com](https://medium.com/@thomasadorme/ten-years-of-ai-in-review-1234567890)

IA : Aspects historiques post 2023 : accélération



Source: [Thomas A Dorfe: "Ten Years of AI in Review", medium.com](#)

Intelligence Artificielles ?



Terme historiquement "mal choisi" ^a

~> le sens actuel reste ambigu : de nombreuses définitions (contradictaires) existent selon les périodes et les auteurs...

^a utilisé la première fois en 1956 par [John McCarthy](#), chercheur à Stanford, lors de la conférence de Dartmouth

Intelligence Artificielles ?



Terme historiquement "mal choisi" ^a

↪ le sens actuel reste ambigu : de nombreuses définitions (contradictaires) existent selon les périodes et les auteurs...

^a utilisé la première fois en 1956 par [John McCarthy](#), chercheur à Stanford, lors de la conférence de Dartmouth

- *"...the science of making computers do things that require intelligence when done by humans."* [Alan Turing, 1940](#)
- *"the field of study that gives computers the ability to learn without being explicitly programmed."* [Arthur Samuel, 1960](#)
- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ."* [Tom Mitchell, 1997](#)
- Notion of *intelligent agent, rational agent*
"...agent that acts in such a way as to reach the best solution or, in an uncertain environment, the best predictable solution."
[Stuart Russel, Peter Norvig, "Intelligence Artificielle" 2015](#)

Intelligences Artificielles ?

Strong / General AI (IA forte / générale)

Weak / Narrow AI (IA faible / spécialisée)

Multimodal AI (IA multi-modale)

Intelligences Artificielles ?

Strong / General AI (IA forte / générale)

- Système IA qui **pensent comme** les êtres humains, avec la capacité de raisonner en général.
- Tente aussi d'expliquer comment les êtres humains pensent.
- nous n'en sommes pas encore là ?

Weak / Narrow AI (IA faible / spécialisée)

Multimodal AI (IA multi-modale)

Intelligences Artificielles ?

Strong / General AI (IA forte / générale)

Weak / Narrow AI (IA faible / spécialisée)

- Système IA qui semble **se comporter comme** des être humains.
- Système IA conçu pour des tâches spécifiques.
- Ne renseigne pas sur la façon dont les êtres humains pensent.
- **Nous en sommes déjà là...** nous l'utilisons tous les jours !
(anti-spam, reconnaissance voix/faciale, traduction...)

Multimodal AI (IA multi-modale)

Intelligences Artificielles ?

Strong / General AI (IA forte / générale)

Weak / Narrow AI (IA faible / spécialisée)

Multimodal AI (IA multi-modale)

- Système IA conçu pour traiter des entrées de nature multiple (texte, images, sons...).

Intelligences Artificielles ?



Machine Learning: un champ prometteur de l'IA

Source [medium.com/machine-learning-for-humans/...](https://medium.com/machine-learning-for-humans/)

Machine learning $\subseteq$ artificial intelligence

ARTIFICIAL INTELLIGENCE

Design an intelligent agent that perceives its environment and makes decisions to maximize chances of achieving its goal.
Subfields: vision, robotics, machine learning, natural language processing, planning, ...

MACHINE LEARNING

Gives "computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)

SUPERVISED LEARNING

Classification, regression

UNSUPERVISED LEARNING

Clustering, dimensionality
reduction, recommendation

REINFORCEMENT LEARNING

Reward maximization

Branches du *Machine Learning*

Entraînement Supervisé (*Supervised learning*)

Un jeu de **données labellisées** est utilisé pour l'entraînement :

- **Classification**

- Classification d'images
- Détection d'objets dans des images
- Reconnaissance de la parole...

- **Régression**

- Prévission de valeur...

- **Détection d'anomalie**

- Anti-spam
- Reconnaissance de défauts (appris)
- Prévission météo...

- ...

Branches du *Machine Learning*

Entraînement Supervisé (*Supervised learning*)

Un jeu de **données labellisées** est utilisé pour l'entraînement :

- **Classification**

- Classification d'images
- Détection d'objets dans des images
- Reconnaissance de la parole...

- **Régression**

- Prévision de valeur...

- **Détection d'anomalie**

- Anti-spam
- Reconnaissance de défauts (appris)
- Prévision météo...

- ...

Branches du *Machine Learning*

Entraînement Supervisé (*Supervised learning*)

Un jeu de **données labellisées** est utilisé pour l'entraînement :

- **Classification**

- Classification d'images
- Détection d'objets dans des images
- Reconnaissance de la parole...

- **Régression**

- Prévission de valeur...

- **Détection d'anomalie**

- Anti-spam
- Reconnaissance de défauts (appris)
- Prévission météo...

- ...

Branches du *Machine Learning*

Entraînement Supervisé (*Supervised learning*)

Un jeu de **données labellisées** est utilisé pour l'entraînement :

- **Classification**

- Classification d'images
- Détection d'objets dans des images
- Reconnaissance de la parole...

- **Régression**

- Prévission de valeur...

- **Détection d'anomalie**

- Anti-spam
- Reconnaissance de défauts (appris)
- Prévission météo...

- ...

Branches du *Machine Learning*

Entraînement non-supervisé (*Unsupervised learning*)

Analyse et groupement de **données non-labellisées**:

- **Groupeement**

- Exploration de données (*Data mining*)
- Regroupement de données WEB
- Analyse de marché
- Analyse de données astronomiques...

- **Détection d'anomalie**

- Fabrication : détection de défauts (même nouveaux)
- Surveillance d'activité : fraude, défaillance, hacking
- Détection de *fake account* sur Internet...

- **Réduction de dimensionnalité**

- Compression de données...

- ...

Branches du *Machine Learning*

Entraînement non-supervisé (*Unsupervised learning*)

Analyse et groupement de **données non-labellisées**:

● Groupement

- Exploration de données (*Data mining*)
- Regroupement de données WEB
- Analyse de marché
- Analyse de données astronomiques...

● Détection d'anomalie

- Fabrication : détection de défauts (même nouveaux)
- Surveillance d'activité : fraude, défaillance, hacking
- Détection de *fake account* sur Internet...

● Réduction de dimensionnalité

- Compression de données...

● ...

Branches du *Machine Learning*

Entraînement non-supervisé (*Unsupervised learning*)

Analyse et groupement de **données non-labellisées**:

● Groupement

- Exploration de données (*Data mining*)
- Regroupement de données WEB
- Analyse de marché
- Analyse de données astronomiques...

● Détection d'anomalie

- Fabrication : détection de défauts (même nouveaux)
- Surveillance d'activité : fraude, défaillance, hacking
- Détection de *fake account* sur Internet...

● Réduction de dimensionnalité

- Compression de données...

● ...

Branches du *Machine Learning*

Entraînement non-supervisé (*Unsupervised learning*)

Analyse et groupement de **données non-labellisées**:

- **Groupeement**

- Exploration de données (*Data mining*)
- Regroupement de données WEB
- Analyse de marché
- Analyse de données astronomiques...

- **Détection d'anomalie**

- Fabrication : détection de défauts (même nouveaux)
- Surveillance d'activité : fraude, défaillance, hacking
- Détection de *fake account* sur Internet...

- **Réduction de dimensionnalité**

- Compression de données...

- ...

Branches du *Machine Learning*

Entraînement par renforcement Deep Reinforcement Learning (DRL)

Un *agent* apprend à piloter un *environment* en maximisant une récompense **reward** :

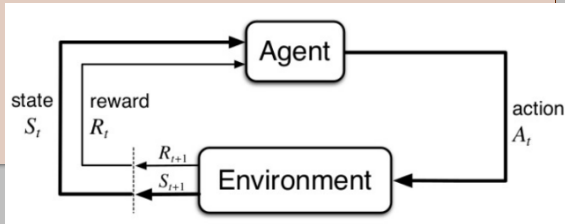
- **Contrôle des systèmes**

- Contrôle de robots, drones
- Optimisation de process de fabrication
- Négociation financière (boursière)...

- **Prise de décision**

- jeux vidéo
- Analyse financière...

- ...



Branches du *Machine Learning*

Entraînement par renforcement Deep Reinforcement Learning (DRL)

Un *agent* apprend à piloter un *environment* en maximisant une *récompense reward* :

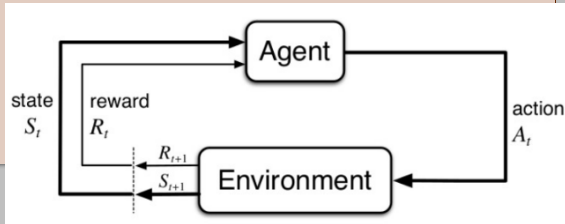
- **Contrôle des systèmes**

- Contrôle de robots, drones
- Optimisation de process de fabrication
- Négociation financière (boursière)...

- **Prise de décision**

- jeux vidéo
- Analyse financière...

- ...



Branches du *Machine Learning*

Entraînement par renforcement Deep Reinforcement Learning (DRL)

Un *agent* apprend à piloter un *environment* en maximisant une *récompense reward* :

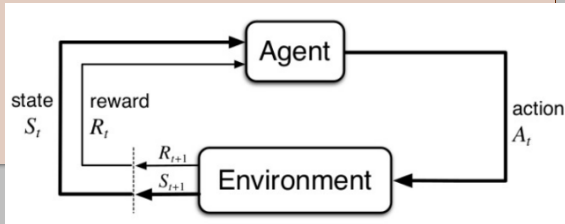
- **Contrôle des systèmes**

- Contrôle de robots, drones
- Optimisation de process de fabrication
- Négociation financière (boursière)...

- **Prise de décision**

- jeux vidéo
- Analyse financière...

- ...



Différents algorithmes du *Machine Learning*

Apprentissage Supervisé :

- Réseau de neurones
- Inférence Bayésienne
- Random forest
- Decision Tree
- Support Vector Machine (SVM)
- K-Nearest Neighbor
- Régression Linéaire
- Régression Logistique...

Apprentissage non-supervisé :

- Réseau de neurones
- Principal Composant Analysis
- Singular Value Decomposition
- K-mean & Prob. clustering...

DRL :

- Réseau de neurones (Q-learning, Actor-Critic, DDPG, PPO...)
- Monte Carlo
- SARSA...

Différents algorithmes du *Machine Learning*

Apprentissage Supervisé :

- Réseau de neurones
- Inférence Bayésienne
- Random forest
- Decision Tree
- Support Vector Machine (SVM)
- K-Nearest Neighbor
- Régression Linéaire
- Régression Logistique...

Apprentissage non-supervisé :

- Réseau de neurones
- Principal Composant Analysis
- Singular Value Decomposition
- K-mean & Prob. clustering...

DRL :

- Réseau de neurones
- Monte Carlo
- SARSA...

Différents algorithmes du *Machine Learning*

Apprentissage Supervisé :

- Réseau de neurones
- Inférence Bayésienne
- Random forest
- Decision Tree
- Support Vector Machine (SVM)
- K-Nearest Neighbor
- Régression Linéaire
- Régression Logistique...

Apprentissage non-supervisé :

- Réseau de neurones
- Principal Composant Analysis
- Singular Value Decomposition
- K-mean & Prob. clustering...

DRL :

- Réseau de neurones
- Monte Carlo
- SARSA...

Grande variété de domaines $\leadsto$ succès
des **Réseaux de Neurones** artificiels

Applications du *Machine Learning*

Vision par ordinateur

- Classification d'images
- Détection d'objects
- Segmentation (sémantique)
- Génération d'images
 - Les 23 meilleurs générateurs d'images par IA (Gratuits et Pros)
- Estimation de pose
- Transfer de style
- Reconnaissance optique de caractères
(Optical Character Recognition: OCR)
- ...

Applications du *Machine Learning*

Vision par ordinateur

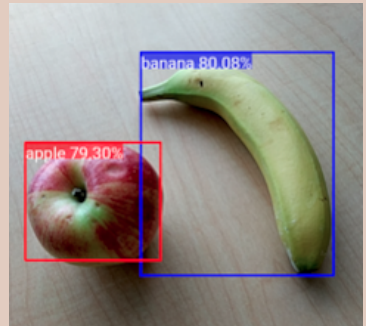
- Classification d'images
 - Détection d'objects
 - Segmentation (sémantique)
 - Génération d'images
 - Les 23 meilleurs générateurs d'images par IA (Gratuits et Pros)
 - Estimation de pose
 - Transfer de style
 - Reconnaissance optique de caractères (Optical Character Recognition: OCR)
 - ...



Applications du *Machine Learning*

Vision par ordinateur

- Classification d'images
- Détection d'objects
- Segmentation (sémantique)
- Génération d'images
- Les 23 meilleurs générateurs d'images par IA (Gratuits et Pros)
- Estimation de pose
- Transfer de style
- Reconnaissance optique de caractères (Optical Character Recognition: OCR)
- ...



source : [Tensorflow](#)

Applications du *Machine Learning*

Vision par ordinateur

- Classification d'images
- Détection d'objects
- Segmentation (sémantique)
- Génération
- Les 23 meilleurs g
- Estimation
- Transfer d
- Reconna
- (Optical Cr
- ...



source : [Tensorflow](#)

Applications du *Machine Learning*

Vision par ordinateur

- Classification d'images
- Détection d'objects
- Segmentation (sémantique)
- Génération d'images

Les 23 meilleurs générateurs d'images par IA (Gratuits et Pros)

- Estimation de pose
- Transfer de style
- Reconnaissance optique de caractères
(Optical Character Recognition: OCR)
- ...



source : [stylegan](#)

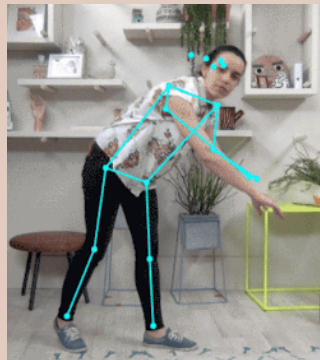
Applications du *Machine Learning*

Vision par ordinateur

- Classification d'images
- Détection d'objects
- Segmentation (sémantique)
- Génération d'images

Les 23 meilleurs générateurs d'images par IA (Gratuits et Pros)

- Estimation de pose
- Transfer de style
- Reconnaissance optique de caractères
(Optical Character Recognition: OCR)
- ...



source : [Tensorflow](#)

Applications du *Machine Learning*

Vision par ordinateur

- Clas
- Déte
- Seg
- Gén

Les 23

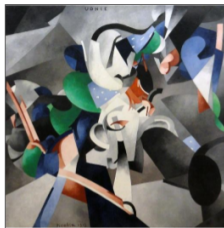
- Estil
- Transfer de style

- Reconnaissance optique de caractères
(Optical Character Recognition: OCR)

- ...



Content



Style



Pastiche

source : [Tensorflow](#)

Applications du *Machine Learning*

Vision par ordinateur

- Classification d'images
- Détection d'objects
- Segmentation (sémantique)
- Génération d'images
 - Les 23 meilleurs générateurs d'images par IA (Gratuits et Pros)
- Estimation de pose
- Transfer de style
- Reconnaissance optique de caractères (Optical Character Recognition: OCR)
- ...



source : [Tensorflow](#)

Applications du *Machine Learning*

Traitement du langage naturel (*Natural Language Processing*)

- Natural Language Understanding (NLU)
- Reconnaissance de la parole (*Speech recognition*)
- Natural Language Generation (NLG)
- Synthèse de la parole (*Speech Synthesis, Text To Speech*)
- Traduction de langues (*Machine Translation (languages)*)
- Agent Conversationnel (*LLM ChatBots*)
- ...

Applications du *Machine Learning*

Traitement du langage naturel (*Natural Language Processing*)

- Natural Language Understanding (NLU)
- Reconnaissance de la parole (*Speech recognition*)
- Natural Language Generation (NLG)
- Synthèse de la parole (*Speech Synthesis, Text To Speech*)
- Traduction de langues (*Machine Translation (languages)*)
- Agent Conversationnel (*LLM ChatBots*)
- ...

Applications du *Machine Learning*

Traitement du langage naturel (*Natural Language Processing*)

- Natural Language Understanding (NLU)
- Reconnaissance de la parole (*Speech recognition*)
- Natural Language Generation (NLG)
- Synthèse de la parole (*Speech Synthesis, Text To Speech*)
- Traduction de langues (*Machine Translation (languages)*)
- Agent Conversationnel (*LLM ChatBots*)
- ...

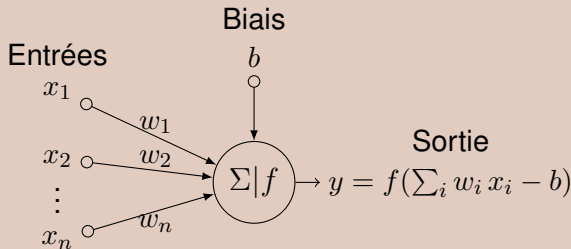
Applications du *Machine Learning*

Traitement du langage naturel (*Natural Language Processing*)

- Natural Language Understanding (NLU)
- Reconnaissance de la parole (*Speech recognition*)
- Natural Language Generation (NLG)
- Synthèse de la parole (*Speech Synthesis, Text To Speech*)
- Traduction de langues (*Machine Translation (languages)*)
- Agent Conversationnel (*LLM ChatBots*)
- ...

Aspects informatiques

Le modèle du neurone artificiel

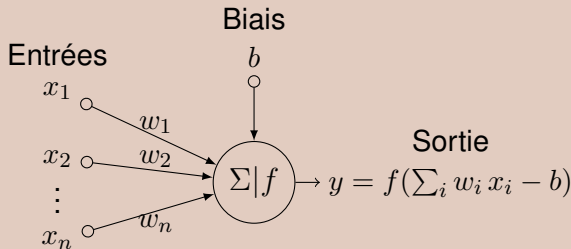


Le **neurone artificiel**:

- reçoit les **entrées** $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$
- calcule la **somme pondérée** de ses entrées : $\sum_i w_i x_i - b$
- donne en **sortie** son activation : $f(\sum_i w_i x_i - b)$
calculée avec sa **activation function** f .

Aspects informatiques

Le modèle du neurone artificiel

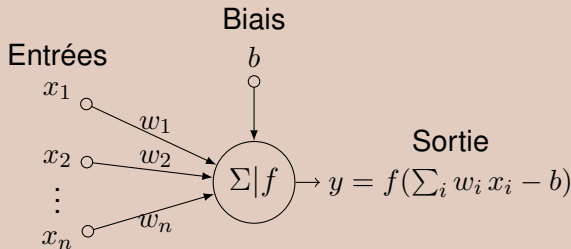


Le **neurone artificiel**:

- reçoit les **entrées** $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$
- calcule la **somme pondérée** de ses entrées : $\sum_i w_i x_i - b$
- donne en **sortie** son activation : $f(\sum_i w_i x_i - b)$
calculée avec sa **activation function** f .

Aspects informatiques

Le modèle du neurone artificiel

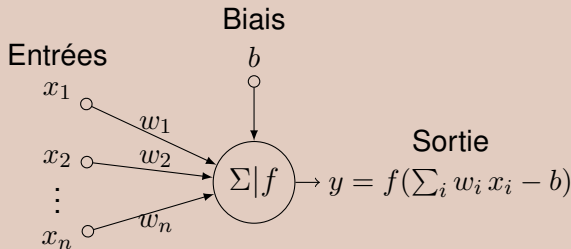


Le **neurone artificiel**:

- reçoit les **entrées** $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$
- calcule la **somme pondérée** de ses entrées : $\sum_i w_i x_i - b$
- donne en **sortie** son activation : $f(\sum_i w_i x_i - b)$
calculée avec sa **activation function** f .

Aspects informatiques

Le modèle du neurone artificiel

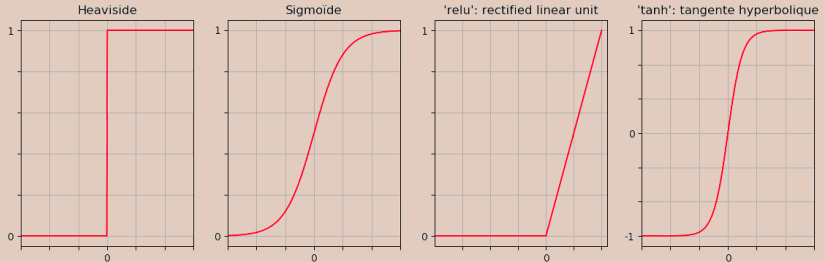


Le **neurone artificiel**:

- reçoit les **entrées** $(x_i)_{i=1..n}$ affectées des **poids** $(w_i)_{i=1..n}$
- calcule la **somme pondérée** de ses entrées : $\sum_i w_i x_i - b$
- donne en **sortie** son activation : $f(\sum_i w_i x_i - b)$ calculée avec sa **activation function** f .

Aspects informatiques

Exemples de fonctions d'activation courantes

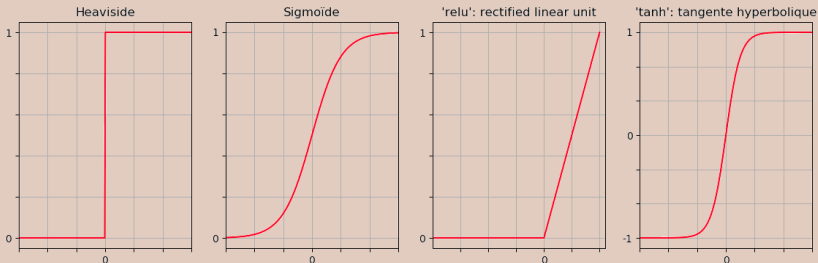


La **fonction d'activation**:

- Introduit un **comportement non-linéaire**, indispensable pour la réussite de l'apprentissage.
- Fixe la plage de sortie du neurone : $[-1, 1]$, $[0, 1]$, $[0, \infty[...$

Aspects informatiques

Exemples de fonctions d'activation courantes

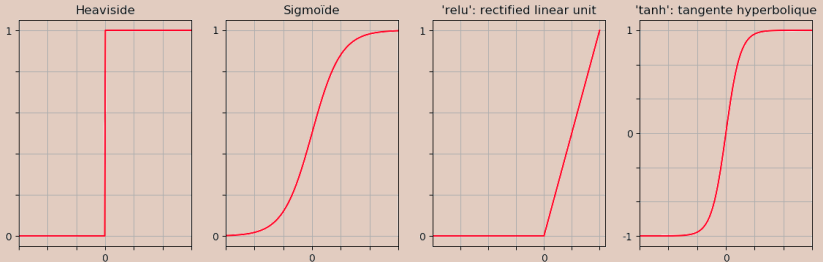


La **fonction d'activation**:

- Introduit un **comportement non-linéaire**, indispensable pour la réussite de l'apprentissage.
- Fixe la plage de sortie du neurone : $[-1, 1]$, $[0, 1]$, $[0, \infty[...$

Aspects informatiques

Exemples de fonctions d'activation courantes



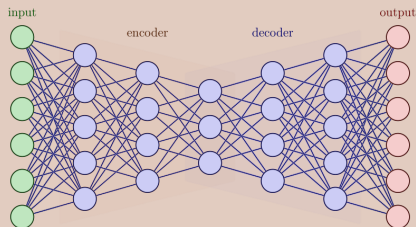
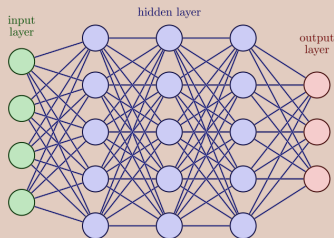
La **fonction d'activation**:

- Introduit un **comportement non-linéaire**, indispensable pour la réussite de l'apprentissage.
- Fixe la plage de sortie du neurone : $[-1, 1]$, $[0, 1]$, $[0, \infty[...$

Aspects informatiques

Réseau de neurones artificiels

Les neurones sont regroupés en couches pour former un **Réseaux de Neurones** artificiels (RN)



Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- *Dense Neural Network*
- *Convolutional (CNN)*
- *Recurrent (RNN)*
- *Auto Encoder (AEN)*
- *Generative Adversarial (GAN)*
- *Transformers*
- *Large Language Model (LLM)*

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- ***Dense Neural Network***

La plus simple des architectures faite de couches successives de neurones, utilisant les algorithmes *Feed Forward* et *Back Propagation*

- *Convolutional (CNN)*
- *Recurrent (RNN)*
- *Auto Encoder (AEN)*
- *Generative Adversarial (GAN)*
- *Transformers*
- *Large Language Model (LLM)*

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- **Dense Neural Network**
- **Convolutional (CNN)** - Réseau de Neurones Convolutionnel
Principalement utilisé pour la classification d'images.
- *Recurrent (RNN)*
- *Auto Encoder (AEN)*
- *Generative Adversarial (GAN)*
- *Transformers*
- *Large Language Model (LLM)*

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- **Dense Neural Network**
- **Convolutional** (CNN)
- **Recurrent** (RNN) - Réseau de Neurones Récurent
Traitement des **séries temporelles** (Exemple: *Long Short-Term Memory (LSTM)*).
- *Auto Encoder (AEN)*
- *Generative Adversarial (GAN)*
- *Transformers*
- *Large Language Model (LLM)*

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- **Dense Neural Network**
- **Convolutional (CNN)**
- **Recurrent (RNN)**
- **Auto Encoder (AEN)**- Réseau de Neurones Auto-encodeur
Réduction de dimensionnalité, compression, débruitage, détection d'anomalie...
- *Generative Adversarial (GAN)*
- *Transformers*
- *Large Language Model (LLM)*

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- **Dense Neural Network**
- **Convolutional** (CNN)
- **Recurrent** (RNN)
- **Auto Encoder** (AEN)
- **Generative Adversarial** (GAN)
génération de texte, images, musique...
- *Transformers*
- *Large Language Model* (LLM)

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- **Dense Neural Network**
- **Convolutional** (CNN)
- **Recurrent** (RNN)
- **Auto Encoder** (AEN)
- **Generative Adversarial** (GAN)
- **Transformers**
(2017) Traitement du langage naturel, puis aussi pour la classification d'images.
- **Large Language Model** (LLM)

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- **Dense Neural Network**
- **Convolutional** (CNN)
- **Recurrent** (RNN)
- **Auto Encoder** (AEN)
- **Generative Adversarial** (GAN)
- **Transformers**
- **Large Language Model** (LLM)
lecture de texte, sons, images... génération de texte, livres,
images, parole, musique (Exemple: *ChatGPT*, *LLama*...)

[Graphique synthétique animé : Saul Dobilas sur Medium]

Architecture des Réseaux de Neurones

Architectures dédiées à des applications spécifiques

- ***Dense Neural Network***
- ***Convolutional*** (CNN)
- ***Recurrent*** (RNN)
- ***Auto Encoder*** (AEN)
- ***Generative Adversarial*** (GAN)
- ***Transformers***
- ***Large Language Model*** (LLM)

[Graphique synthétique animé : [Saul Dobilas sur Medium](#)]

Enjeux sociétaux de l'IA : **Explicabilité**

devenue rapidement une priorité dans la recherche :

→ **xAI** : **Explicable** *Artificial Intelligence*

→ **iML** : **Interpretable** *Machine learning*

Explicabilité des Réseaux de Neurones (RN)

- **Inexplicabilité** des résultats calculés par les RN → obstacle à leur diffusion.
- L'apprentissage profond avec les RN souvent dénigré comme une "boîte noire" par les scientifiques ayant une "approche cartésienne"...
- La complexité grandissante des RN (LLM par exemple) rend l'explication simple de leurs décisions extrêmement difficile.

Enjeux sociétaux de l'IA : **Explicabilité**

devenue rapidement une priorité dans la recherche :

→ **xAI** : **Explicable** *Artificial Intelligence*

→ **iML** : **Interpretable** *Machine learning*

Explicabilité des Réseaux de Neurones (RN)

- **Inexplicabilité** des résultats calculés par les RN → obstacle à leur diffusion.
- L'apprentissage profond avec les RN souvent dénigré comme une "boîte noire" par les scientifiques ayant une "approche cartésienne"...
- La complexité grandissante des RN (LLM par exemple) rend l'explication simple de leurs décisions extrêmement difficile.

Enjeux sociétaux de l'IA : **Explicabilité**

devenue rapidement une priorité dans la recherche :

→ **xAI** : **Explicable** *Artificial Intelligence*

→ **iML** : **Interpretable** *Machine learning*

Explicabilité des Réseaux de Neurones (RN)

- **Inexplicabilité** des résultats calculés par les RN → obstacle à leur diffusion.
- L'apprentissage profond avec les RN souvent dénigré comme une "boîte noire" par les scientifiques ayant une "approche cartésienne"...
- La complexité grandissante des RN (LLM par exemple) rend l'explication simple de leurs décisions extrêmement difficile.

Enjeux sociétaux de l'IA : **Prise de décision**

Prise de décision

- Un nombre croissant de décisions dans des domaines sensibles (justice pénale, santé, assurance, défense...) confiées aux algorithmes ML source d'inquiétude...
- Les algorithmes de prise de décision reposent inévitablement sur des hypothèses (qualité des données d'apprentissage par exemple) souvent difficiles à vérifier.
- Se pose la question de la transparence des algorithmes, des données d'entraînement... (*Open Source*).

Enjeux sociétaux de l'IA : Prise de décision

Prise de décision

- Un nombre croissant de décisions dans des domaines sensibles (justice pénale, santé, assurance, défense...) confiées aux algorithmes ML source d'inquiétude...
- Les algorithmes de prise de décision reposent inévitablement sur des hypothèses (qualité des données d'apprentissage par exemple) souvent difficiles à vérifier.
- Se pose la question de la transparence des algorithmes, des données d'entraînement... (*Open Source*).

Enjeux sociétaux de l'IA : Prise de décision

Prise de décision

- Un nombre croissant de décisions dans des domaines sensibles (justice pénale, santé, assurance, défense...) confiées aux algorithmes ML source d'inquiétude...
- Les algorithmes de prise de décision reposent inévitablement sur des hypothèses (qualité des données d'apprentissage par exemple) souvent difficiles à vérifier.
- Se pose la question de la transparence des algorithmes, des données d'entraînement... (*Open Source*).

Enjeux sociétaux de l'IA : **Certification**

Certification

- **Évaluation et Certification** des systèmes IA
~> enjeu majeur pour leur intégration dans l'industrie.
- **Certification** formelle des algorithmes de ML
~> reste aujourd'hui un sujet de recherche :
 - LNE : Certification des processus pour l'IA
 - Fraunhofer : Audit et certification des systèmes d'IA

Enjeux sociétaux de l'IA : Certification

Certification

- **Évaluation et Certification** des systèmes IA
→ enjeu majeur pour leur intégration dans l'industrie.
- **Certification** formelle des algorithmes de ML
→ reste aujourd'hui un sujet de recherche :
 - LNE : Certification des processus pour l'IA
 - Fraunhofer : Audit et certification des systèmes d'IA

Études : entraînement d'un réseau de neurones YOLO pour la détection de petits objets 3D



Livrables: projet GiHb [UCIA_ObjectDetection](#)

Détection d'objets 3D

Deux études menées en novembre-décembre 2024:

Étude préliminaire (v1)

Objectifs :

- Faisabilité d'entraîner un réseau YOLO à détecter les petits objets 3D du projet UCIA
- Tester l'exploitation sur RPi4 d'un réseau YOLO entraîné.

Détection d'objets 3D

Deux études menées en novembre-décembre 2024:

Étude préliminaire (v1)

Objectifs :

- Faisabilité d'entraîner un réseau YOLO à détecter les petits objets 3D du projet UCIA
- Tester l'exploitation sur RPi4 d'un réseau YOLO entraîné.

Détection d'objets 3D

Deux études menées en novembre-décembre 2024:

Étude préliminaire (v1)

Objectifs :

- Faisabilité d'entraîner un réseau YOLO à détecter les petits objets 3D du projet UCIA
- Tester l'exploitation sur RPi4 d'un réseau YOLO entraîné.

Configuration :

- Caméra Raspberry standard montée sur trépied
- Entraînement des réseaux sur PC + carte graphique Nvidia
- Exploitation sur RPi4 4Go RAM des réseaux entraînés.

Détection d'ob

Deux études r

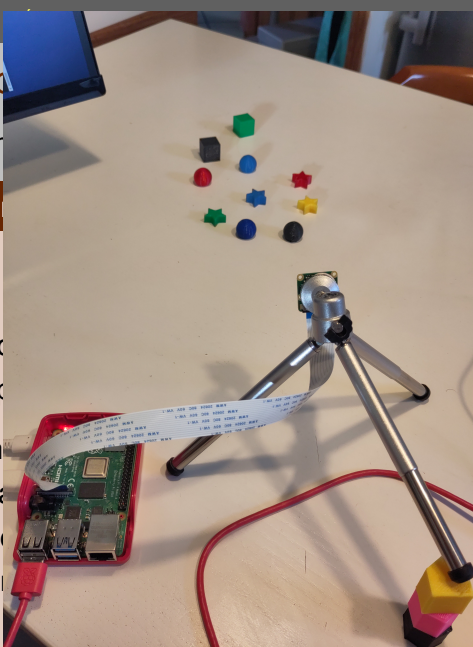
Étude prélimi

Objectifs :

- Faisabilité
objets 3D c
- Tester l'exp

Configuration

- Caméra Ra
- Entraînem
- Exploitation



er les petits

entraîné.

rique Nvidia
traînés.

Détection d'objets 3D

Étude consolidée (v2)

Objectifs :

- Consolidation de l'entraînement de réseaux YOLO dans les conditions matérielles du CDC du projet UCIA
- Visualisation à distance (navigateur WEB, bureau à distance)
- Prototypage Python du déplacement du Thymio.

Détection d'objets 3D

Étude consolidée (v2)

Objectifs :

- Consolidation de l'entraînement de réseaux YOLO dans les conditions matérielles du CDC du projet UCIA
- Visualisation à distance (navigateur WEB, bureau à distance)
- Prototypage Python du déplacement du Thymio.

Détection d'objets 3D

Étude consolidée (v2)

Objectifs :

- Consolidation de l'entraînement de réseaux YOLO dans les conditions matérielles du CDC du projet UCIA
- Visualisation à distance (navigateur WEB, bureau à distance)
- Prototypage Python du déplacement du Thymio.

Détection d'objets 3D

Étude consolidée (v2)

Objectifs :

- Consolidation de l'entraînement de réseaux YOLO dans les conditions matérielles du CDC du projet UCIA
- Visualisation à distance (navigateur WEB, bureau à distance)
- Prototypage Python du déplacement du Thymio.

Configuration :

- Caméra Raspberry grand angle, fixée sur le support Thymio.
- Entraînement des réseaux sur PC + carte graphique Nvidia
- Exploitation sur RPi4 4Go RAM fixée sur le support Thymio.

Détection d'objets 3D

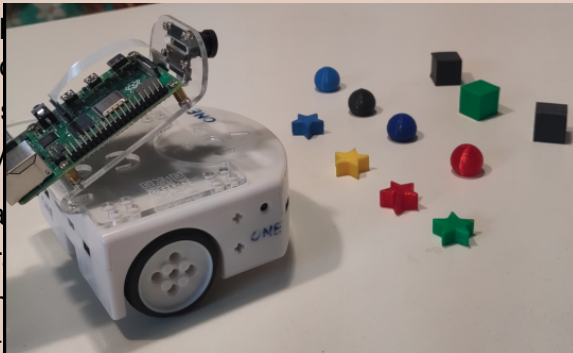
Étude consolidée (v2)

Objectifs :

- Conso
- Visualis
- Prototy

Configura

- Camér
- Entraîn
- Exploit



dans les

(distance)

t Thymio.

e Nvidia

Thymio.

Détection d'objets 3D : Banque d'images annotées

Images de l'étude préliminaire (caméra Standard)

Caméra Raspberry Standard, 62 images, 10 objets/images

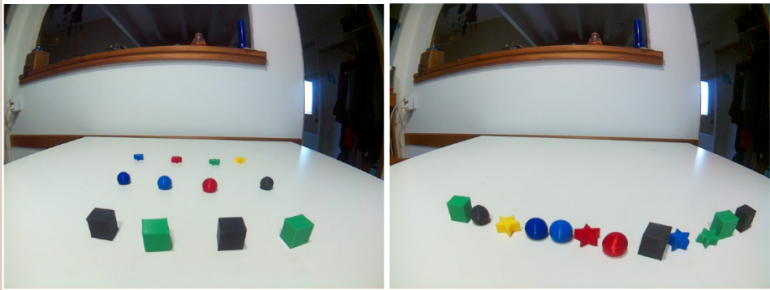
→ 620 objets



Détection d'objets 3D : Banque d'images annotées

Images de l'étude consolidée (caméra Grand Angle)

Caméra Raspberry Grand Angle, 200 images, 12 objets/images
↪ 2400 objets



Détection d'objets 3D : Banque d'images annotées

Annotation des images (v1)

Images annotées sur le site Roboflow

→ projet [ucia-ia-object-detection](#)

The screenshot displays the Roboflow web interface for a dataset. At the top, a search bar indicates 'Search 500,000+ Open Source Computer Vision Projects...'. The left sidebar contains navigation options: 'Object Detection', 'Overview', 'Images' (62), 'Dataset' (selected, with a green badge), 'Model' (0), 'API Docs', and 'Analytics'. The main content area shows '62 Total Images' with a 'View All Images →' link and a row of five image thumbnails. Below this, the 'Dataset Split' section shows three categories: 'TRAIN SET' (50 Images, 81%), 'VALID SET' (10 Images, 16%), and 'TEST SET' (2 Images, 3%). The 'Preprocessing' section lists 'Auto-Orient: Applied', 'Resize: Stretch to 640x640', and 'Grayscale: Applied'. The 'Augmentations' section states 'No augmentations were applied.'

Détection d'objets 3D : Banque d'images annotées

Annotation des images (v2)

Images annotées sur le site Roboflow

↪ projet [ucia-ia-object-detection-v2.0](#)

Search 500,000+ Open Source Computer Vision Projects...

Object Detection

Overview

Images 202

Dataset 0

Model 0

</> API Docs

Analytics

202 Total Images [View All Images →](#)

Dataset Split

TRAIN SET 81% 163 Images	VALID SET 19% 38 Images	TEST SET 0% 1 Images
-----------------------------	----------------------------	-------------------------

Preprocessing

Auto-Orient: Applied
Resize: Stretch to 640x640
Grayscale: Applied

Augmentations

No augmentations were applied.

Détection d'objets 3D : réseau de neurones YOLO

Réseau de neurones **YOLO** (*You Only Look Once*)

- Réseau de neurones populaire : **détection d'objets, segmentation, détection pose...**
- Lancé en 2015, très utilisé (rapidité et précision).
- Réseau **pré-entraîné** avec les images du WEB :
 - Détection, Segmentation & Pose ~ MS-COCO (328000 images, 80 classes d'objets)
 - Classification ~ ImageNet (plus d'un million d'images)
- Les versions de YOLO sont sur le site Ultralytics : docs.ultralytics.com/fr

Détection d'objets 3D : réseau de neurones YOLO

Réseau de neurones **YOLO** (*You Only Look Once*)

- Réseau de neurones populaire : **détection d'objets, segmentation, détection pose...**
- Lancé en 2015, très utilisé (rapidité et précision).
- Réseau **pré-entraîné** avec les images du WEB :
 - Détection, Segmentation & Pose $\leadsto$ **MS-COCO** (328000 images, 80 classes d'objets)
 - Classification $\leadsto$ ImageNet (plus d'un million d'images)
- Les versions de YOLO sont sur le site Ultralytics : docs.ultralytics.com/fr

Détection d'objets 3D : réseau de neurones YOLO

Réseau de neurones **YOLO** (*You Only Look Once*)

- Réseau de neurones populaire : **détection d'objets, segmentation, détection pose...**
- Lancé en 2015, très utilisé (rapidité et précision).
- Réseau **pré-entraîné** avec les images du WEB :
 - Détection, Segmentation & Pose $\leadsto$ **MS-COCO** (328000 images, 80 classes d'objets)
 - Classification $\leadsto$ ImageNet (plus d'un million d'images)
- Les versions de YOLO sont sur le site Ultralytics : docs.ultralytics.com/fr

Détection d'objets 3D : réseau de neurones YOLO

Versions du réseau YOLO retenues pour l'étude

- **YOLOv8** : version aboutie, stable, parfaitement connue.
- **YOLO11** : pour les dernières innovations.

Détection d'objets 3D : réseau de neurones YOLO

Versions du réseau YOLO retenues pour l'étude

- **YOLOv8** : version aboutie, stable, parfaitement connue.
- **YOLO11** : pour les dernières innovations.
- 5 versions de complexité croissante **n, s, m, l** et **x** :

YOLO...n	→ nano	pour les tâches petites et légères.
YOLO...s	→ small	mise à niveau de nano, meilleure précision.
YOLO...m	→ medium	pour une utilisation à usage général.
YOLO...l	→ large	meilleure précision au prix d'un calcul plus lourd.
YOLO...x	→ Extra-large	pour une précision et des performances maximales.

Détection d'objets 3D : réseau de neurones YOLO

Versions du réseau YOLO retenues pour l'étude

- **YOLOv8** : version aboutie, stable, parfaitement connue.
- **YOLO11** : pour les dernières innovations.
- 5 versions de complexité croissante **n, s, m, l** et **x** :

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Détection d'objets 3D : réseau de neurones YOLO

Versions du réseau YOLO retenues pour l'étude

- **YOLOv8** : version aboutie, stable, parfaitement connue.
- **YOLO11** : pour les dernières innovations.
- 5 versions de complexité croissante **n**, **s**, **m**, **l** et **x** :

YOLO...n	→ nano	pour les tâches petites et légères.
YOLO...s	→ small	mise à niveau de nano, meilleure précision.
YOLO...m	→ medium	pour une utilisation à usage général.
YOLO...l	→ large	meilleure précision au prix d'un calcul plus lourd.
YOLO...x	→ Extra-large	pour une précision et des performances maximales.

~> Les versions **n** et **s** sont retenues (cible RPi4)

Détection d'objets 3D : réseau de neurones YOLO

Versions du réseau YOLO retenues pour l'étude

- **YOLOv8** : version aboutie, stable, parfaitement connue.
- **YOLO11** : pour les dernières innovations.
- 5 versions de complexité croissante **n**, **s**, **m**, **l** et **x** :

YOLO...n	→ nano	pour les tâches petites et légères.
YOLO...s	→ small	mise à niveau de nano, meilleure précision.
YOLO...m	→ medium	pour une utilisation à usage général.
YOLO...l	→ large	meilleure précision au prix d'un calcul plus lourd.
YOLO...x	→ Extra-large	pour une précision et des performances maximales.

~> Les versions **n** et **s** sont retenues (cible RPi4)

~> 4 réseaux utilisés

YOLOv8n, **YOLOv8s**, **YOLO11n**, **YOLO11s**

Détection d'objets 3D : entraînement du réseau YOLO

Choix des méta-paramètres d'entraînement

Méta-paramètres **batch** et **epochs**

étude v1

Paramètre	Description	Plage de valeurs
epochs	Nombre de répétitions du processus complet d'entraînement pour converger vers le meilleur état de réseau entraîné	20, 40, 60, 80, 100
batch	Nombre d'images fournies dans un lot d'images d'entraînement	2, 4, 8, 10, 16, 20, 30

étude v2

Paramètre	Description	Plage de valeurs
epochs	Nombre de répétitions du processus complet d'entraînement pour converger vers le meilleur état de réseau entraîné	20, 40, 60, 80, 100
batch	Nombre d'images fournies dans un lot d'images d'entraînement	2, 4, 8, 16, 32

Détection d'objets 3D : entraînement du réseau YOLO

Choix des paramètres d'entraînement

Paramètres **imgz**, **pretrained** et **seed**:

Paramètre	Description	Valeur
imgz	Taille des image (en pixels)	640
patience	Nombre d'époques à attendre sans amélioration des mesures de validation avant d'arrêter l'entraînement. Permet d'éviter le sur-entraînement en arrêtant le processus lorsque les performances atteignent un plateau.	100
pretrained	Détermine s'il faut utiliser un modèle pré-entraîné.	True
seed	Définit la graine aléatoire pour l'entraînement pour garantir la reproductibilité des résultats d'une exécution à l'autre avec les mêmes configurations.	1234
workers	Nombre de threads de travail pour le chargement des données.	0

Détection d'objets 3D : entraînement du réseau YOLO

La combinaison des méta-paramètres donne de nombreux entraînements à stocker :

$4 \times 7 \times 5 = 140$ entraînements pour l'étude V1, 100 pour l'étude V2.

Détection d'objets 3D : entraînement du réseau YOLO

La combinaison des méta-paramètres donne de nombreux entraînements à stocker :

$4 \times 7 \times 5 = 140$ entraînements pour l'étude V1, 100 pour l'étude V2.

Nommage des dossiers d'entraînement (exemple V2)

Training/YOLO-trained-V2/UCIA-YOLOvvv/batch-BB_epo-EEE

vvv $\leadsto$ version du réseau YOLO : (v8n, v8s, 11n, 11s)

BB $\leadsto$ méta-paramètre **batch** : (02, 04, 08, 16, 32)

EEE $\leadsto$ méta-paramètre **epochs** : (020, 040, 060, 080, 100)

Détection d'objets 3D : entraînement du réseau YOLO

La combinaison des méta-paramètres donne de nombreux entraînements à stocker :

$4 \times 7 \times 5 = 140$ entraînements pour l'étude V1, 100 pour l'étude V2.

Nommage des dossiers d'entraînement (exemple V2)

`Training/YOLO-trained-V2/UCIA-YOLOvvv/batch-BB_epo-EEE`

vvv $\leadsto$ version du réseau YOLO : (v8n, v8s, 11n, 11s)

BB $\leadsto$ méta-paramètre **batch** : (02, 04, 08, 16, 32)

EEE $\leadsto$ méta-paramètre **epochs** : (020, 040, 060, 080, 100)

Programmes Python d'entraînement des réseaux YOLO

`train_YOLOv8.py` et `train_YOLO11.py` développés pour l'étude V1, complétés pour V2.

Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement **UCIA-YOLOvvv/batch-BB_epo-EEE/**

- Extraits des images d'entraînement
- Résultats avec les images de validation
- Fichier `confusion_matrix.png` : matrice de confusion
- Fichier `results.png` : tracé des statistiques et métriques d'entraînement

Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement **UCIA-YOLOvvv/batch-BB_epo-EEE/**

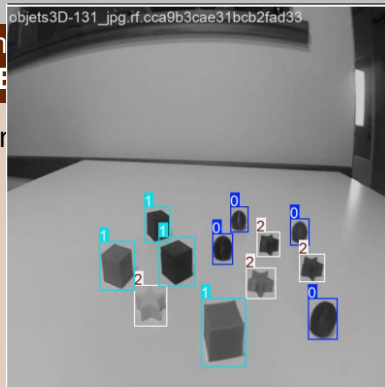
- Extraits des images d'entraînement
- Résultats avec les images de validation
- Fichier `confusion_matrix.png` : matrice de confusion
- Fichier `results.png` : tracé des statistiques et métriques d'entraînement

Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement
UCIA-YOLOvvv/batch-BB_epo-EEB

- Extraits des images d'entraînement



- Résultats avec les images de validation
- Fichier `confusion_matrix.png` : matrice de confusion

Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement **UCIA-YOLOvvv/batch-BB_epo-EEE/**

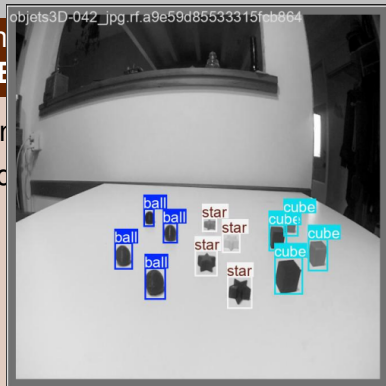
- Extraits des images d'entraînement
- Résultats avec les images de validation
- Fichier `confusion_matrix.png` : matrice de confusion
- Fichier `results.png` : tracé des statistiques et métriques d'entraînement

Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement **UCIA-YOLOvvv/batch-BB_epo-EEB**

- Extraits des images d'entraînement
- Résultats avec les images de validation



- Fichier `confusion_matrix.png` : matrice de confusion
- Fichier `results.png` : tracé des statistiques et métriques

Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement **UCIA-YOLOvvv/batch-BB_epo-EEE/**

- Extraits des images d'entraînement
- Résultats avec les images de validation
- Fichier **confusion_matrix.png** : matrice de confusion
- Fichier **results.png** : tracé des statistiques et métriques d'entraînement

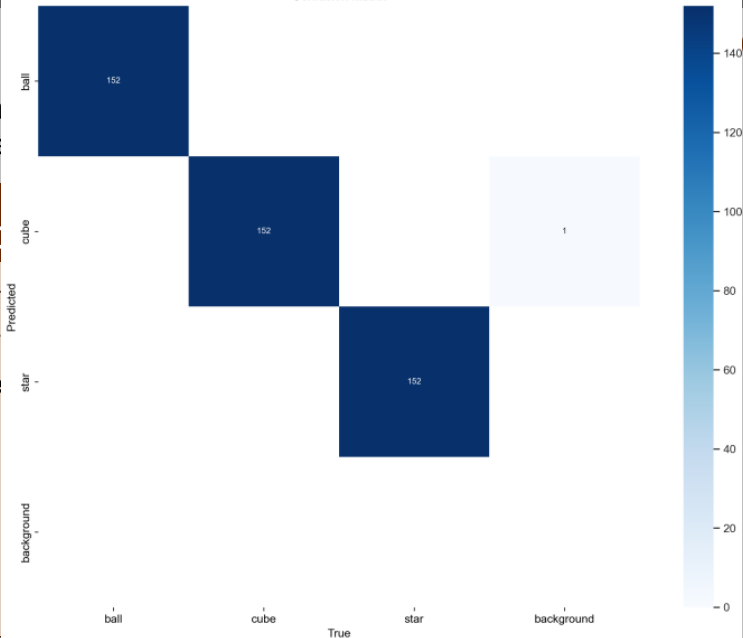
Détection

Les entra
graphique

Contenu
UCIA-YO

- Extraire
- Résul
- Fichier

Confusion Matrix



Détection d'objets 3D : entraînement du réseau YOLO

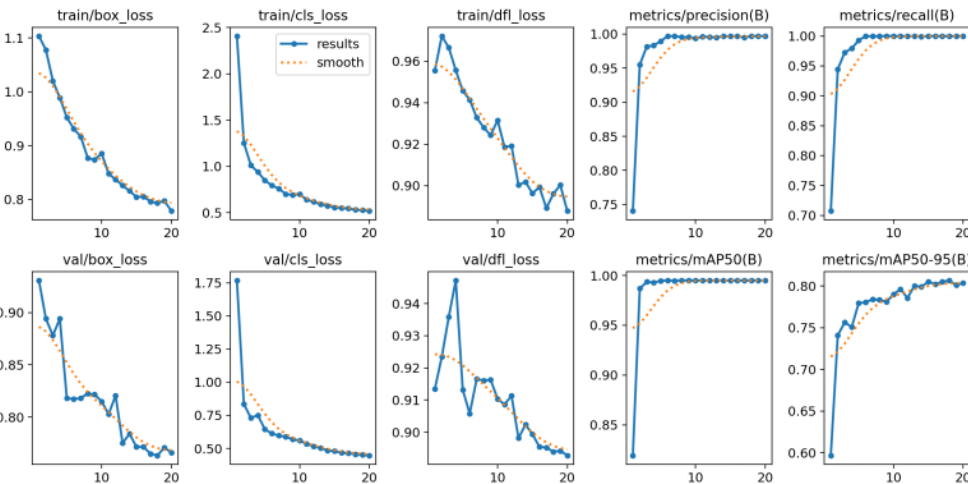
Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement **UCIA-YOLOvvv/batch-BB_epo-EEE/**

- Extraits des images d'entraînement
- Résultats avec les images de validation
- Fichier `confusion_matrix.png` : matrice de confusion
- Fichier `results.png` : tracé des statistiques et métriques d'entraînement

Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte



Détection d'objets 3D : entraînement du réseau YOLO

Les entraînements sont faits sur un PC équipé d'une carte graphique **Nvidia Quadro TRX8000**.

Contenu des dossiers d'entraînement **UCIA-YOLOvvv/batch-BB_epo-EEE/**

- Extraits des images d'entraînement
- Résultats avec les images de validation
- Fichier `confusion_matrix.png` : matrice de confusion
- Fichier `results.png` : tracé des statistiques et métriques d'entraînement

Détection d'objets 3D : entraînement du réseau YOLO

Contenu des dossiers des poids

UCIA-YOLOvvv/batch-BB_epo-EEE/weights

- **best.pt** : fichier binaire des poids du réseau entraîné (au format **pytorch**).
- Formats optimisés générés pour RPi4 :
- Taille des fichiers binaires des poids des réseaux YOLO

Détection d'objets 3D : entraînement du réseau YOLO

Contenu des dossiers des poids

UCIA-YOLOvvv/batch-BB_epo-EEE/weights

- **best.pt** : fichier binaire des poids du réseau entraîné (au format **pytorch**).
- Formats optimisés générés pour RPi4 :
 - **best.onnx** : poids du réseau entraîné au format **ONNX**
 - **best.ncmm** : poids du réseau entraîné au format **NCNN**
- Taille des fichiers binaires des poids des réseaux YOLO

Détection d'objets 3D : entraînement du réseau YOLO

Contenu des dossiers des poids

UCIA-YOLOvvv/batch-BB_epo-EEE/weights

- **best.pt** : fichier binaire des poids du réseau entraîné (au format **pytorch**).
- Formats optimisés générés pour RPi4 :
 - **best.onnx** : poids du réseau entraîné au format **ONNX**
 - **best.ncnn** : poids du réseau entraîné au format **NCNN**
- Taille des fichiers binaires des poids des réseaux YOLO

	yolov8n	yolov8s	yolo11n	yolo11s
.pt	~6 Mo	~22 Mo	~6 Mo	~19 Mo
.onnx	~12 Mo	~43 Mo	~11 Mo	~37 Mo
.ncnn	~12 Mo	~43 Mo	~11 Mo	~37 Mo

Détection d'objets 3D : Performances entraînements

- Le module Python **ultralytics** fournit des fonctions pour évaluer les indicateurs de performance de détection d'objets.

Détection d'objets 3D : Performances entraînements

- Le module Python **ultralytics** fournit des fonctions pour évaluer les indicateurs de performance de détection d'objets.
- Programmes [eval_YOLOv8.py](#) et [eval_YOLO11.py](#) développés (V1 et V2) pour évaluer les différentes versions d'entraînement des réseaux YOLO.

Détection d'objets 3D : Performances entraînements

- Le module Python **ultralytics** fournit des fonctions pour évaluer les indicateurs de performance de détection d'objets.
- Programmes `eval_YOLOv8.py` et `eval_YOLO11.py` développés (V1 et V2) pour évaluer les différentes versions d'entraînement des réseaux YOLO.
- ↪ 4 fichiers résultats (exemple pour V2):
`results_yolov8n-V2.txt`, `results_yolov8s-V2.txt`,
`results_yolo11n-V2.txt`, `results_yolo11s-V2.txt`,
contenant chacun les valeurs des indicateurs de performance pour les différentes valeurs des méta-paramètres.

Détection d'objets 3D : Performances entraînements

Temps moyen d'inférence

Pour info, sur le PC de calcul (carte *Nvidia Quadro TRX8000*)

Réseau	Temps moyen d'inférence[ms]
yolov8n	2
yolo11n	2
yolov8s	4
yolo11s	4

→ Architecture **s** deux fois plus lente que **n**

Détection d'objets 3D : Performances entraînements

Indicateurs de précision détection des réseaux détecteur

`process_results.py` $\leadsto$ synthèse des évaluation des réseaux entraînés.

- recall : capacité du réseau à identifier toutes les objets dans les images.
- mAP50 : précision moyenne pour un seuil d'*intersection sur union* IoU de 0.5 (détectons "faciles").
- mAP50-95 : précision moyenne pour différents seuils IoU allant de 0.5 à 0.95 (différents niveaux de difficulté de détection).
- fitness = $0.1 * \text{mAP50} + 0.9 * \text{mAP50-95}$

Détection d'objets 3D : Performances entraînements

Indicateurs de précision détection des réseaux détecteur

`process_results.py` $\leadsto$ synthèse des évaluation des réseaux entraînés.

- **recall** : capacité du réseau à identifier toutes les objets dans les images.
- mAP50 : précision moyenne pour un seuil d'*intersection sur union* IoU de 0.5 (détectons "faciles").
- mAP50-95 : précision moyenne pour différents seuils **IoU** allant de 0.5 à 0.95 (différents niveaux de difficulté de détection).
- $\text{fitness} = 0.1 * \text{mAP50} + 0.9 * \text{mAP50-95}$

Détection d'objets 3D : Performances entraînements

Indicateurs de précision détection des réseaux détecteur

`process_results.py` $\leadsto$ synthèse des évaluation des réseaux entraînés.

- **recall** : capacité du réseau à identifier toutes les objets dans les images.
- **mAP50** : précision moyenne pour un seuil d'*intersection sur union* **IoU** de 0.5 (détections "faciles").
- mAP50-95 : précision moyenne pour différents seuils **IoU** allant de 0.5 à 0.95 (différents niveaux de difficulté de détection).
- $\text{fitness} = 0.1 * \text{mAP50} + 0.9 * \text{mAP50-95}$

Détection d'objets 3D : Performances entraînements

Indicateurs de précision détection des réseaux détecteur

`process_results.py` $\leadsto$ synthèse des évaluation des réseaux entraînés.

- -
- objets dans les
section sur union

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



- mAP50-95 : précision moyenne pour différents seuils IoU allant de

Détection d'objets 3D : Performances entraînements

Indicateurs de précision détection des réseaux détecteur

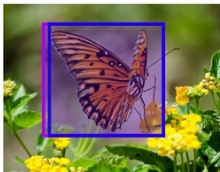
`process_results.py` $\leadsto$ synthèse des évaluation des réseaux entraînés.



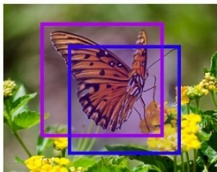
Excellent

Good

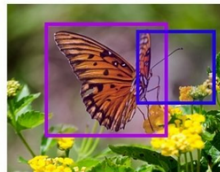
Poor



IoU = 0.971



IoU = 0.772



IoU = 0.324



mAP50-95 : précision moyenne pour différents seuils IoU allant de 0.5 à 0.95 (différents niveaux de difficulté de détection).



$\text{fitness} = 0.1 * \text{mAP50} + 0.9 * \text{mAP50-95}$

Détection d'objets 3D : Performances entraînements

Indicateurs de précision détection des réseaux détecteur

`process_results.py` $\leadsto$ synthèse des évaluation des réseaux entraînés.

- **recall** : capacité du réseau à identifier toutes les objets dans les images.
- **mAP50** : précision moyenne pour un seuil d'*intersection sur union* **IoU** de 0.5 (détections "faciles").
- **mAP50-95** : précision moyenne pour différents seuils **IoU** allant de 0.5 à 0.95 (différents niveaux de difficulté de détection).
- $\text{fitness} = 0.1 * \text{mAP50} + 0.9 * \text{mAP50-95}$

Détection d'objets 3D : Performances entraînements

Indicateurs de précision détection des réseaux détecteur

`process_results.py` $\leadsto$ synthèse des évaluation des réseaux entraînés.

- **recall** : capacité du réseau à identifier toutes les objets dans les images.
- **mAP50** : précision moyenne pour un seuil d'*intersection sur union* **IoU** de 0.5 (détections "faciles").
- **mAP50-95** : précision moyenne pour différents seuils **IoU** allant de 0.5 à 0.95 (différents niveaux de difficulté de détection).
- **fitness** = $0.1 * \text{mAP50} + 0.9 * \text{mAP50-95}$

Dét

File <Training/Results/results_yolov8n-V2.txt>

#meta-params	recall	mAP50-95	fitness
batch-02_epo-060	1.0	0.821	0.838
batch-02_epo-080	1.0	0.821	0.838
batch-04_epo-100	1.0	0.821	0.838
batch-04_epo-080	1.0	0.819	0.836

File <Training/Results/results_yolo11n-V2.txt>

#meta-params	recall	mAP50-95	fitness
batch-08_epo-100	1.0	0.814	0.833
batch-04_epo-080	1.0	0.813	0.831
batch-16_epo-100	1.0	0.813	0.831
batch-04_epo-060	1.0	0.809	0.828

File <Training/Results/results_yolov8s-V2.txt>

#meta-params	recall	mAP50-95	fitness
batch-02_epo-080	1.0	0.837	0.853
batch-08_epo-060	1.0	0.837	0.852
batch-02_epo-060	1.0	0.835	0.851
batch-02_epo-100	1.0	0.834	0.850

File <Training/Results/results_yolo11s-V2.txt>

#meta-params	recall	mAP50-95	fitness
batch-04_epo-080	1.0	0.839	0.855
batch-32_epo-060	1.0	0.838	0.854
batch-08_epo-080	1.0	0.837	0.852
batch-02_epo-100	1.0	0.836	0.852

traînements

détecteur

des réseaux en-

objets dans les

section sur union

ouils **IoU** allant de
(ction).

Détection d'objets 3D : Performances entraînements

Configurations d'entraînement à tester sur RPi4

Meilleures performance :

- UCIA-YOLOv8n/batch-04_epo-100
- UCIA-YOLOv8s/batch-02_epo-080
- UCIA-YOLO11n/batch-02_epo-100
- UCIA-YOLO11s/batch-04_epo-080

Détection d'objets 3D : Performances entraînements

Configurations d'entraînement à tester sur RPi4

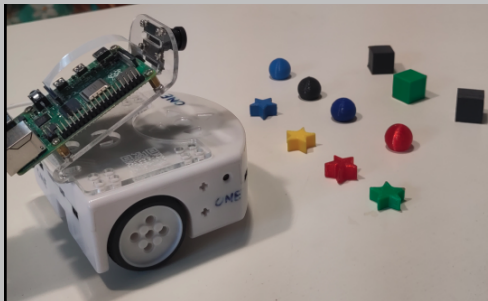
Meilleures performance :

- UCIA-YOLOv8n/batch-04_epo-100
- UCIA-YOLOv8s/batch-02_epo-080
- UCIA-YOLO11n/batch-02_epo-100
- UCIA-YOLO11s/batch-04_epo-080

On retrouve des tendances connues :

- **batch_size** petit $\leadsto$ favorise en entraînement de qualité,
- **epochs** élevé $\leadsto$ compense le petit nombre d'images fournies à chaque entraînement.

Étude : Exploitation sur RPi4 des réseaux entraînés YOLO selon le CDC UCIA



Configuration RPi4-UCIA

Configuration RPi4-UCIA

- Carte micro-SD rapide de 64 GB
- Flash carte micro-SD :
système d'exploitation **Raspberry PI OS (64-bit)**
- Point d'accès Wifi [SSID : RPi4-UCIA, clef :poppy!station]
- Compte : **ucia**, mdp : **poppy!station**
ouvert automatiquement au démarrage de la RPi4
- Environnement Virtuel Python (EVP) **vision**
activé automatiquement à l'ouverture du compte **ucia**
- Arborescence des réseaux YOLO entraînés copiée sur la
carte micro-SD.

Configuration RPi4-UCIA

Configuration RPi4-UCIA

- Carte micro-SD rapide de 64 GB
- Flash carte micro-SD :
système d'exploitation **Raspberry PI OS (64-bit)**
- Point d'accès Wifi [SSID : RPi4-UCIA, clef :poppy!station]
- Compte : `ucia`, mdp : `poppy!station`
ouvert automatiquement au démarrage de la RPi4
- Environnement Virtuel Python (EVP) `vision`
activé automatiquement à l'ouverture du compte `ucia`
- Arborescence des réseaux YOLO entraînés copiée sur la
carte micro-SD.

Configuration RPi4-UCIA

Co

Raspberry Pi Imager v1.8.5

Système d'exploitation



Raspberry Pi OS (64-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop (Recommended)
(Recommended)
Publié le : 2024-11-19
Mis en cache sur votre ordinateur



Raspberry Pi OS (32-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop
Publié le : 2024-11-19
En ligne - 1.1 GO à télécharger



Raspberry Pi OS (Legacy, 32-bit)
A port of Debian Bullseye with security updates and desktop environment
Publié le : 2024-10-22
En ligne - 0.9 GO à télécharger

- Point d'accès Wifi [SSID : RPi4-UCIA, clef :poppy!station]
- Compte : ucia, mdp : poppy!station

ouvert automatiquement au démarrage de la RPi4

JLC – 2025/01/06

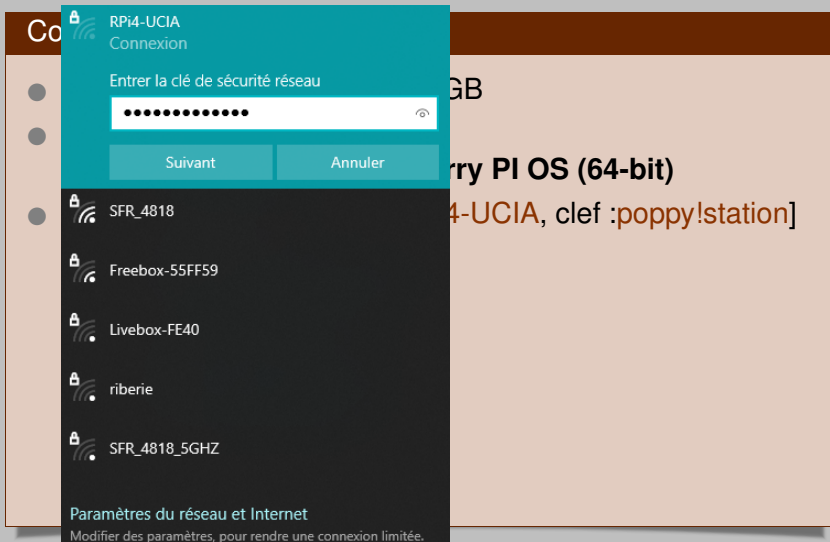
41/49

Configuration RPi4-UCIA

Configuration RPi4-UCIA

- Carte micro-SD rapide de 64 GB
- Flash carte micro-SD :
système d'exploitation **Raspberry PI OS (64-bit)**
- Point d'accès Wifi [SSID : **RPi4-UCIA**, clef : **poppy!station**]
- Compte : **ucia**, mdp : **poppy!station**
ouvert automatiquement au démarrage de la RPi4
- Environnement Virtuel Python (EVP) **vision**
activé automatiquement à l'ouverture du compte **ucia**
- Arborescence des réseaux YOLO entraînés copiée sur la
carte micro-SD.

Configuration RPi4-UCIA



Configuration RPi4-UCIA

Configuration RPi4-UCIA

- Carte micro-SD rapide de 64 GB
- Flash carte micro-SD :
système d'exploitation **Raspberry PI OS (64-bit)**
- Point d'accès Wifi [SSID : **RPi4-UCIA**, clef : **poppy!station**]
- Compte : **ucia**, mdp : **poppy!station**
ouvert automatiquement au démarrage de la RPi4
- Environnement Virtuel Python (EVP) **vision**
activé automatiquement à l'ouverture du compte **ucia**
- Arborescence des réseaux YOLO entraînés copiée sur la
carte micro-SD.

Configuration RPi4-UCIA

Configuration RPi4-UCIA

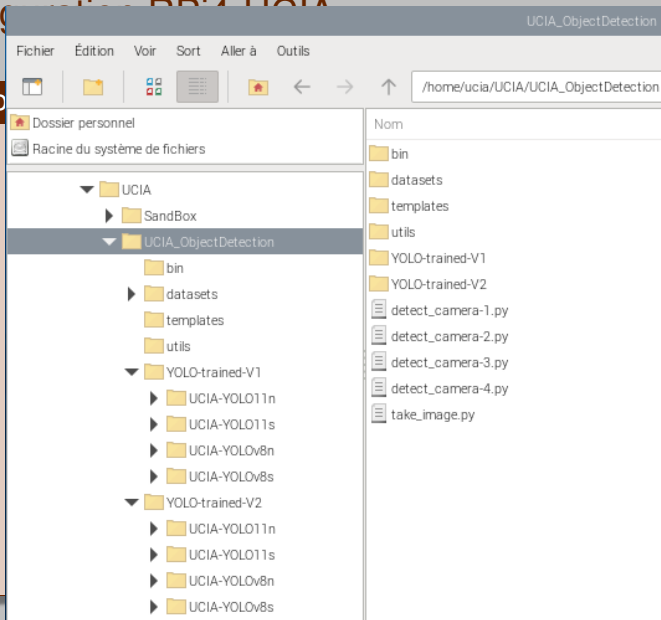
- Carte micro-SD rapide de 64 GB
- Flash carte micro-SD :
système d'exploitation **Raspberry PI OS (64-bit)**
- Point d'accès Wifi [SSID : **RPi4-UCIA**, clef :**poppy!station**]
- Compte : **ucia**, mdp : **poppy!station**
ouvert automatiquement au démarrage de la RPi4
- Environnement Virtuel Python (EVP) **vision**
activé automatiquement à l'ouverture du compte **ucia**
- Arborescence des réseaux YOLO entraînés copiée sur la
carte micro-SD.

Configuration RPi4-UCIA

Configuration RPi4-UCIA

- Carte micro-SD rapide de 64 GB
- Flash carte micro-SD :
système d'exploitation **Raspberry PI OS (64-bit)**
- Point d'accès Wifi [SSID : **RPi4-UCIA**, clef :**poppy!station**]
- Compte : **ucia**, mdp : **poppy!station**
ouvert automatiquement au démarrage de la RPi4
- Environnement Virtuel Python (EVP) **vision**
activé automatiquement à l'ouverture du compte **ucia**
- Arborescence des réseaux YOLO entraînés copiée sur la carte micro-SD.

Configuration RPi4 UOLIA



Configuration RPi4-UCIA

Programmes Python d'exploitation des réseaux YOLO

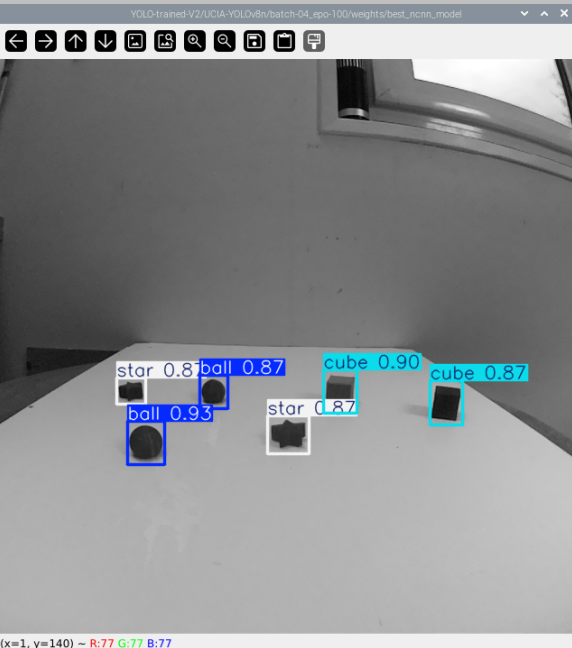
- 4 programmes Python `detect_camera-n.py` (n=1,2,3,4) :
 - `detect_camera-1.py` : images (N&B) des détections du réseau YOLO choisi + infos élémentaires sur les détections
 - `detect_camera-2.py` : images (couleur) des détections du réseau YOLO choisi + infos détaillées sur les détections
 - `detect_camera-3.py` : serveur à l'adresse 10.99.99.1:5000/video en attente de la connexion d'un navigateur WEB sur cette URL.
 - Connexion sur 10.99.99.1:5000/quit ~ terminer l'application
 - Connexion sur 10.99.99.1:5000/halt ~ Shutdown de la carte RPi4 (permet l'extinction)
 - Temps d'inférence du réseau YOLOv8n < 0.5 seconde

Configuration RPi4-UCIA

Programmes Python d'exploitation des réseaux YOLO

- 4 programmes Python `detect_camera-n.py` (n=1,2,3,4) :
- `detect_camera-1.py` : images (N&B) des détections du réseau YOLO choisi + infos élémentaires sur les détections
- `detect_camera-2.py` : images (couleur) des détections du réseau YOLO choisi + infos détaillées sur les détections
- `detect_camera-3.py` : serveur à l'adresse 10.99.99.1:5000/video en attente de la connexion d'un navigateur WEB sur cette URL.
 - Connexion sur 10.99.99.1:5000/quit ~ terminer l'application
 - Connexion sur 10.99.99.1:5000/halt ~ Shutdown de la carte RPi4 (permet l'extinction)
- Temps d'inférence du réseau YOLOv8n < 0.5 seconde

Config



YOLO

,2,3,4) :
ons du réseau
ns

```

ucia@raspberrypi ~/UCIA/UCIA_ObjectDetection
Fichier  Édition  Onglets  Aide

ucia@raspberrypi ~  x  ucia@raspberrypi ...  x

Speed: 18.2ms preprocess, 464.7ms inference, 4.9ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 476.2ms
Speed: 18.7ms preprocess, 476.2ms inference, 6.7ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 476.5ms
Speed: 17.7ms preprocess, 476.5ms inference, 5.2ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 463.0ms
Speed: 12.8ms preprocess, 463.0ms inference, 4.4ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 477.2ms
Speed: 10.2ms preprocess, 477.2ms inference, 4.8ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 466.8ms
Speed: 16.1ms preprocess, 466.8ms inference, 4.6ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 546.0ms
Speed: 19.6ms preprocess, 546.0ms inference, 4.7ms postprocess per image at shape (1, 3, 640, 640)

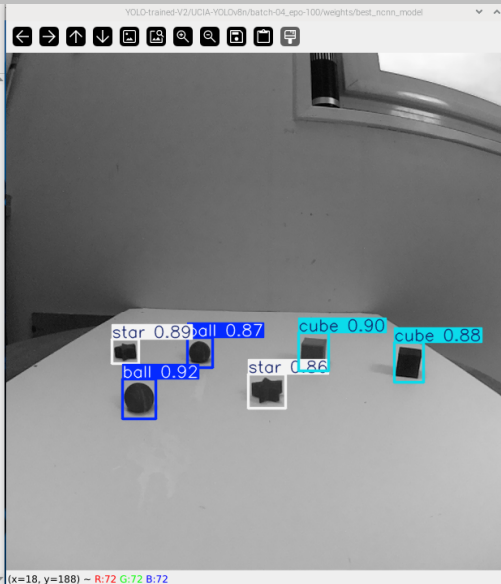
0: 640x640 2 balls, 2 cubes, 2 stars, 464.7ms
Speed: 11.5ms preprocess, 464.7ms inference, 4.9ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 465.6ms
Speed: 13.7ms preprocess, 465.6ms inference, 4.4ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 435.1ms
Speed: 15.8ms preprocess, 435.1ms inference, 4.0ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 2 balls, 2 cubes, 2 stars, 463.9ms
Speed: 16.7ms preprocess, 463.9ms inference, 5.0ms postprocess per image at shape (1, 3, 640, 640)

```



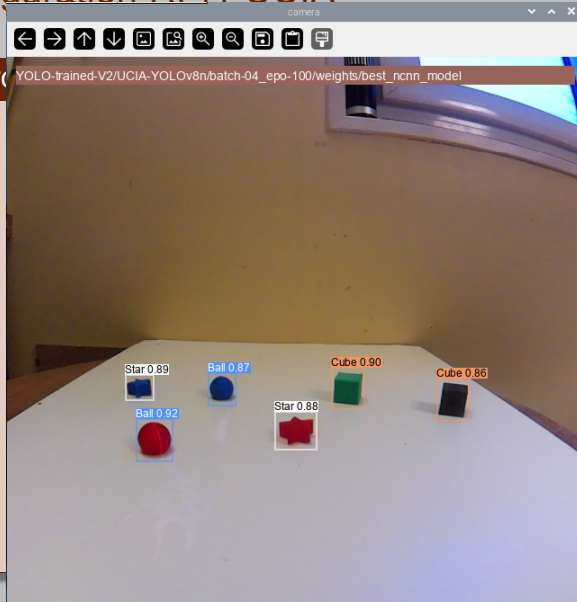
● detect_camera-2.py : images (couleur) des detections du réseau

Configuration RPi4-UCIA

Programmes Python d'exploitation des réseaux YOLO

- 4 programmes Python `detect_camera-n.py` (n=1,2,3,4) :
- `detect_camera-1.py` : images (N&B) des détections du réseau YOLO choisi + infos élémentaires sur les détections
- `detect_camera-2.py` : images (couleur) des détections du réseau YOLO choisi + infos détaillées sur les détections
- `detect_camera-3.py` : serveur à l'adresse 10.99.99.1:5000/video en attente de la connexion d'un navigateur WEB sur cette URL.
 - Connexion sur 10.99.99.1:5000/quit ~ terminer l'application
 - Connexion sur 10.99.99.1:5000/halt ~ Shutdown de la carte RPi4 (permet l'extinction)
- Temps d'inférence du réseau YOLOv8n < 0.5 seconde

Configuration RPi4-UCIA



Pro

YOLO

,2,3,4) :

ons du réseau
ns

ections du réseau

Configuration RPi4-UCIA

```
ucia@raspberrypi: ~/UCIA/UCIA_ObjectDetection
Fichier  Édition  Onglets  Aide

ucia@raspberrypi ~  x  ucia@raspberrypi ~  x

1 87 0 478 407 514 359 27 22 27
2 86 1 297 439 344 396 138 6 27

0: 640x640 2 balls, 2 cubes, 2 stars, 490.4ms
Speed: 17.7ms preprocess, 490.4ms inference, 5.4ms postprocess per image at shape (1, 3, 640, 640)
0 92 1 143 452 184 403 141 2 20
1 90 2 360 393 397 346 12 76 47
2 87 3 130 384 163 355 13 39 98
2 87 0 477 407 514 358 28 22 26
0 87 3 223 390 254 352 8 38 102
2 86 1 297 439 344 396 137 6 26

0: 640x640 2 balls, 2 cubes, 2 stars, 484.4ms
Speed: 10.0ms preprocess, 484.4ms inference, 3.8ms postprocess per image at shape (1, 3, 640, 640)
0 92 1 143 452 184 403 141 2 21
1 90 2 360 394 397 346 11 76 46
0 88 3 223 390 254 352 8 39 102
2 87 1 297 439 344 397 137 4 29
1 87 0 477 407 514 358 27 23 26
2 84 3 131 384 163 355 12 40 97

0: 640x640 2 balls, 2 cubes, 2 stars, 493.6ms
Speed: 17.5ms preprocess, 493.6ms inference, 4.2ms postprocess per image at shape (1, 3, 640, 640)
0 92 1 143 452 184 403 142 2 21
1 90 2 360 394 397 347 11 75 49
2 88 3 130 384 163 355 13 39 97
1 87 0 478 407 514 358 28 23 25
0 87 3 223 389 254 352 7 39 106
2 86 1 297 439 344 396 138 5 29

0: 640x640 2 balls, 2 cubes, 2 stars, 503.4ms
Speed: 11.7ms preprocess, 503.4ms inference, 4.6ms postprocess per image at shape (1, 3, 640, 640)
0 92 1 143 452 184 403 142 2 22
1 90 2 360 394 397 346 11 75 48
2 89 3 130 384 163 354 13 40 98
1 87 0 478 407 514 358 27 23 27
0 87 3 223 389 254 352 7 39 106
2 87 1 297 439 344 396 138 5 26
```



Configuration RPi4-UCIA

Programmes Python d'exploitation des réseaux YOLO

- 4 programmes Python `detect_camera-n.py` (n=1,2,3,4) :
- `detect_camera-1.py` : images (N&B) des détections du réseau YOLO choisi + infos élémentaires sur les détections
- `detect_camera-2.py` : images (couleur) des détections du réseau YOLO choisi + infos détaillées sur les détections
- `detect_camera-3.py` : serveur à l'adresse `10.99.99.1:5000/video` en attente de la connexion d'un navigateur WEB sur cette URL.
 - Connexion sur `10.99.99.1:5000/quit` $\leadsto$ terminer l'application
 - Connexion sur `10.99.99.1:5000/halt` $\leadsto$ Shutdown de la carte RPi4 (permet l'extinction)
- Temps d'inférence du réseau YOLOv8n < 0.5 seconde

Configuration RPi4-UCIA

Pro

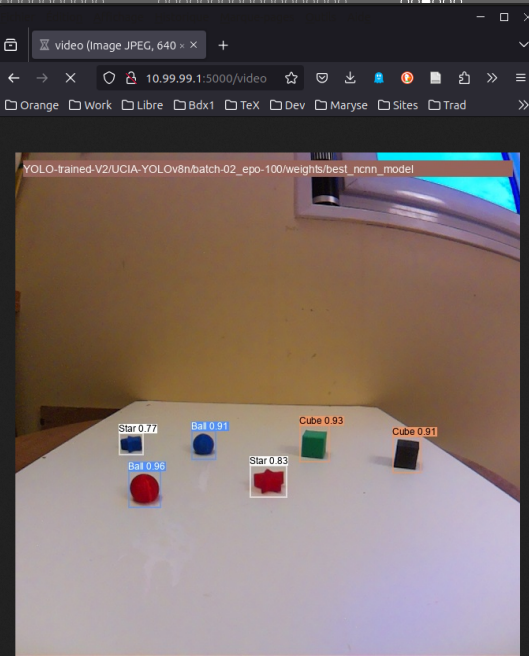
ucia@raspberrypi: ~/UCIA/UCIA_ObjectDetection

Fichier Édition Onglets Aide

```
(vision) ucia@raspberrypi:~ $ cd UCIA/UCIA_ObjectDetection/
(vision) ucia@raspberrypi:~/UCIA/UCIA_ObjectDetection $ python inf_camera-3.py
Inférences du réseau YOLO-trained-V2/UCIA-YOLOv8n/batch-02_epo-100/weights/best_ncnn_model
[2:04:04.699082432] [3012] INFO Camera camera_manager.cpp:325 libcamera v0.3.2+99-1230f78d
[2:04:04.726671307] [3018] WARN RPiSdn sdn.cpp:40 Using legacy SDN tuning - please consider moving SDN inside rpi.denoise
[2:04:04.729170897] [3018] INFO RPI vc4.cpp:447 Registered camera /base/soc/i2c0mux/i2c@1/ov5647@36 to Unicam device /dev/media3 and ISP device /dev/media0
[2:04:04.729470820] [3018] INFO RPI pipeline_base.cpp:1120 Using configuration file '/usr/share/libcamera/pipeline/rpi/vc4/rpi_apps.yaml'
[2:04:04.737281753] [3012] INFO Camera camera.cpp:1197 configuring streams: (0) 640x640 - RGB888 (1) 1296x972-SGBRG10_CSI2P
[2:04:04.739848305] [3018] INFO RPI vc4.cpp:622 Sensor: /base/soc/i2c0mux/i2c@1/ov5647@36 - Selected sensor format: 1296x972-SGBRG10_1X10 - Selected unicam format: 1296x972-pGAA
* Serving Flask app 'inf_camera-3'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.99.99.1:5000
Press CTRL+C to quit
```

● Temps d'inférence du réseau YOLOv8n < 0.5 seconde

Config



aux YOLO

n=1,2,3,4) :

ections du réseau
ctionsdétections du réseau
ns.99.99.1:5000/video
EB sur cette URL.terminer l'application
shutdown de la carte

Configuration RPi4-UCIA

```
Pro ucia@raspberrypi: ~/UCIA/UCIA_ObjectDetection
Fichier  Édition  Onglets  Aide
1 91 2 360 393 398 346 13 76 47
1 91 0 477 407 514 359 30 23 19
2 82 1 297 438 344 397 139 4 23
2 80 3 130 384 163 354 15 38 96

0: 640x640 2 balls, 2 cubes, 2 stars, 422.7ms
Speed: 13.3ms preprocess, 422.7ms inference, 2.8ms postprocess per image at shape (1, 3, 640, 640)
0 95 1 143 451 184 404 143 2 19
1 92 2 360 393 398 346 14 75 46
0 91 3 223 389 254 352 8 37 104
1 91 0 478 407 514 359 30 23 21
2 83 1 297 438 344 397 140 4 23
2 75 3 131 384 163 354 14 39 96

0: 640x640 2 balls, 2 cubes, 2 stars, 420.9ms
Speed: 12.1ms preprocess, 420.9ms inference, 4.3ms postprocess per image at shape (1, 3, 640, 640)
0 95 1 143 451 184 403 143 2 19
1 92 2 360 393 398 346 13 76 48
1 91 0 477 407 514 359 30 23 22
0 89 3 223 389 254 352 10 38 102
2 84 1 297 438 344 397 140 3 24
2 74 3 130 384 162 355 15 39 96
```

● Temps d'inférence du réseau YOLOv8n < 0.5 seconde

Configuration RPi4-UCIA

Programmes Python d'exploitation des réseaux YOLO

- 4 programmes Python `detect_camera-n.py` (n=1,2,3,4) :
- `detect_camera-1.py` : images (N&B) des détections du réseau YOLO choisi + infos élémentaires sur les détections
- `detect_camera-2.py` : images (couleur) des détections du réseau YOLO choisi + infos détaillées sur les détections
- `detect_camera-3.py` : serveur à l'adresse `10.99.99.1:5000/video` en attente de la connexion d'un navigateur WEB sur cette URL.
 - Connexion sur `10.99.99.1:5000/quit` $\leadsto$ terminer l'application
 - Connexion sur `10.99.99.1:5000/halt` $\leadsto$ Shutdown de la carte RPi4 (permet l'extinction)
- Temps d'inférence du réseau YOLOv8n < 0.5 seconde

Configuration RPi4-UCIA

Programmes Python d'exploitation des réseaux YOLO

- 4 programmes Python `detect_camera-n.py` (n=1,2,3,4) :
- `detect_camera-1.py` : images (N&B) des détections du réseau YOLO choisi + infos élémentaires sur les détections
- `detect_camera-2.py` : images (couleur) des détections du réseau YOLO choisi + infos détaillées sur les détections
- `detect_camera-3.py` : serveur à l'adresse **10.99.99.1:5000/video** en attente de la connexion d'un navigateur WEB sur cette URL.
 - Connexion sur **10.99.99.1:5000/quit** $\leadsto$ terminer l'application
 - Connexion sur **10.99.99.1:5000/halt** $\leadsto$ Shutdown de la carte RPi4 (permet l'extinction)
- **Temps d'inférence du réseau YOLOv8n < 0.5 seconde**

Configuration RPi4-UCIA

Exploitation des réseaux YOLO : Bureau à distance (VNC)

- Démarrage RPi4 $\leadsto$ lancement de `detect_camera-3.py` en attente de l'ouverture de `10.99.99.1:5000/video` par un navigateur... **CTRL + C** pour quitter et fermer le terminal.
- Double clic icône [Ai] $\leadsto$ lancement de `detect_camera-3.py` : en attente de l'ouverture de `10.99.99.1:5000/video` par un navigateur... CTRL + C pour quitter et fermer le terminal.
- Connexion sur `10.99.99.1:5000/quit` $\leadsto$ terminer l'application
- Connexion sur `10.99.99.1:5000/halt` $\leadsto$ Shutdown de la carte RPi4 (permet l'extinction)

Configuration RPi4-UCIA

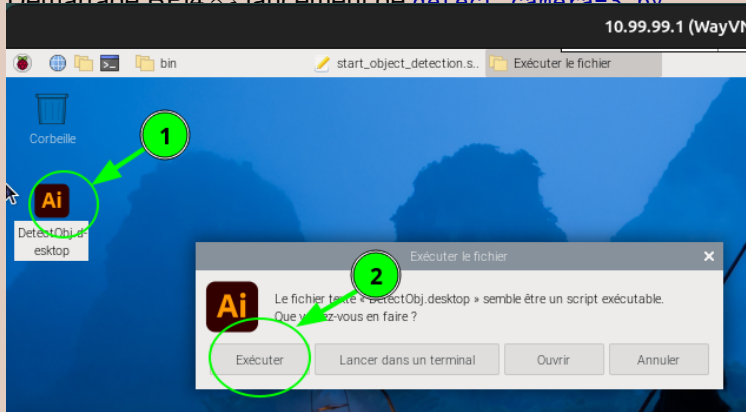
Exploitation des réseaux YOLO : Bureau à distance (VNC)

- Démarrage RPi4 $\leadsto$ lancement de `detect_camera-3.py` en attente de l'ouverture de `10.99.99.1:5000/video` par un navigateur... **CTRL + C** pour quitter et fermer le terminal.
- Double clic icône [Ai] $\leadsto$ lancement de `detect_camera-3.py` : en attente de l'ouverture de `10.99.99.1:5000/video` par un navigateur... **CTRL + C** pour quitter et fermer le terminal.
- Connexion sur `10.99.99.1:5000/quit` $\leadsto$ terminer l'application
- Connexion sur `10.99.99.1:5000/halt` $\leadsto$ Shutdown de la carte RPi4 (permet l'extinction)

Configuration RPi4-UCIA

Exploitation des réseaux YOLO : Bureau à distance (VNC)

- Démarrage RPi4 → lancement de `detect_camera-3.py`



- Connexion sur `10.99.99.1:5000/quit` → terminer l'application

Configuration RPi4-UCIA

Exploitation des réseaux YOLO : Bureau à distance (VNC)

- Démarrage RPi4 $\leadsto$ lancement de `detect_camera-3.py` en attente de l'ouverture de `10.99.99.1:5000/video` par un navigateur... **CTRL + C** pour quitter et fermer le terminal.
- Double clic icône [Ai] $\leadsto$ lancement de `detect_camera-3.py` : en attente de l'ouverture de `10.99.99.1:5000/video` par un navigateur... **CTRL + C** pour quitter et fermer le terminal.
- Connexion sur `10.99.99.1:5000/quit` $\leadsto$ terminer l'application
- Connexion sur `10.99.99.1:5000/halt` $\leadsto$ Shutdown de la carte RPi4 (permet l'extinction)

Configuration RPi4-UCIA

Exploitation des réseaux YOLO : Navigateur web

- Démarrage RPi4 ~> lancement de `detect_camera-3.py` en attente sur `10.99.99.1:5000/video` par un navigateur...
- Connexion navigateur sur `10.99.99.1:5000/video`
~> images (couleur) des détections du réseau YOLO
- Connexion navigateur sur `10.99.99.1:5000/quit`
~> terminer l'application
- Connexion navigateur sur `10.99.99.1:5000/halt`
~> Shutdown de la carte RPi4 (permet l'extinction)

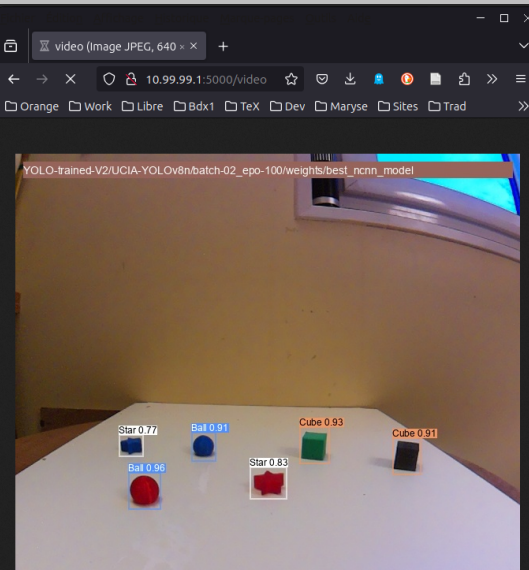
Configuration RPi4-UCIA

Exploitation des réseaux YOLO : Navigateur web

- Démarrage RPi4 ∼ lancement de `detect_camera-3.py` en attente sur `10.99.99.1:5000/video` par un navigateur...
- Connexion navigateur sur `10.99.99.1:5000/video` ∼ images (couleur) des détections du réseau YOLO
- Connexion navigateur sur `10.99.99.1:5000/quit` ∼ terminer l'application
- Connexion navigateur sur `10.99.99.1:5000/halt` ∼ Shutdown de la carte RPi4 (permet l'extinction)

Configuration RPi4-UCIA

Ex



web

mera-3.py
avigateur...

deo
YOLO

Configuration RPi4-UCIA

Exploitation des réseaux YOLO : Navigateur web

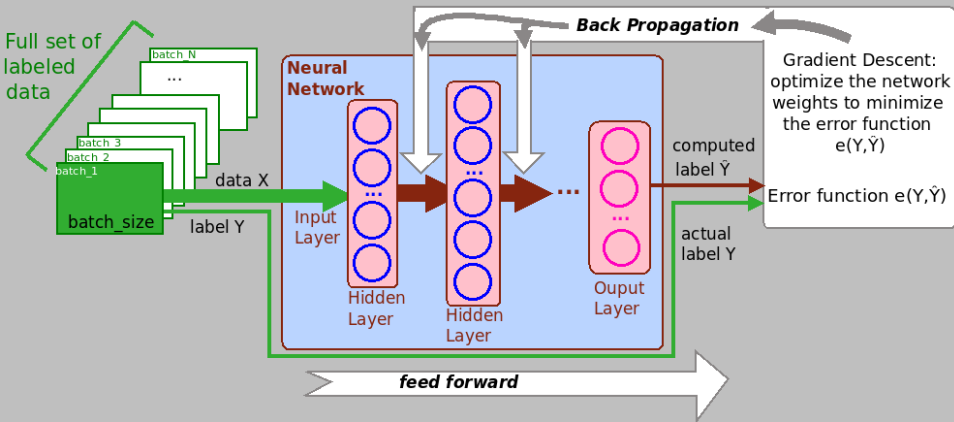
- Démarrage RPi4 ~> lancement de `detect_camera-3.py` en attente sur `10.99.99.1:5000/video` par un navigateur...
- Connexion navigateur sur `10.99.99.1:5000/video`
~> images (couleur) des détections du réseau YOLO
- Connexion navigateur sur `10.99.99.1:5000/quit`
~> terminer l'application
- Connexion navigateur sur `10.99.99.1:5000/halt`
~> Shutdown de la carte RPi4 (permet l'extinction)

References

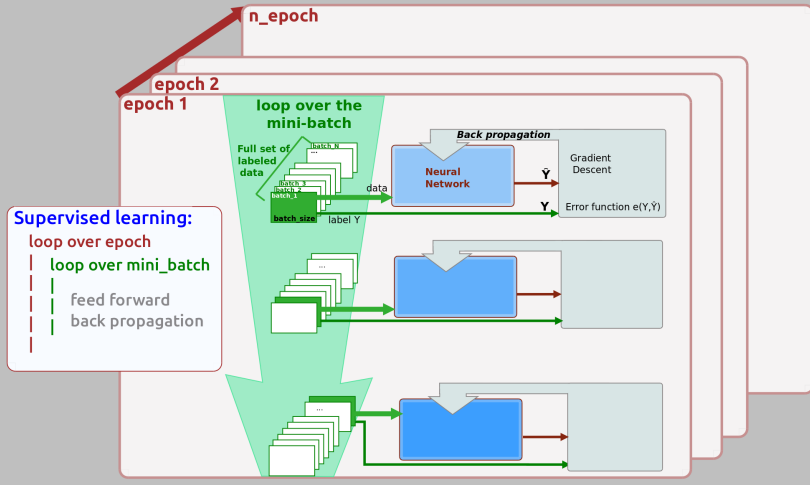
- UCIA Cahier des charges : Robot ROSA avec Intelligence Artificielle OpenSource et OpenHardware
- Page du site roboflow pour l'accès public au jeu de données de l'étude : <https://universe.roboflow.com/ucia/ucia-ia-object-detection/dataset/2>
- «Ultralytics YOLO11: Faster Than You Can Imagine!», Ankan Ghosh, October 8, 2024
<https://learnopencv.com/yolo11/>
- Article Wikipédia sur les indicateurs de précision:
https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel
- «Analyse approfondie des mesures de performance», site Ultralytics
<https://docs.ultralytics.com/fr/guides/yolo-performance-metrics/>
- «The Complete Guide to Object Detection Evaluation Metrics: From IoU to mAP and More»
<https://medium.com/@prathameshamrutkar3/the-complete-guide-to-object-detection-evaluation-metrics-from-iou-to-map-and-more-1a23c0ea3c9d>

Annexes techniques

Supervised learning : Feed Forward and Back Propagation



- The dataset is split into (mini) **batches** of size `batch_size`
- After each *feed forward* the *Back Propagation* algorithm modifies the weights neurons to minimize the error e .



- Training with the whole dataset is repeated **n_epoch** times,
- The network state at the end of epoch **n** becomes the initial state for epoch **n+1**.