FORUM INNOVATION
INGÉNIERIE | INFORMATIQUE |
ENTREPRENEURIAT | UQAR

CÉGEP DE MATANE

Argus service de vision intelligent pour les jeux

par Jérôme Bouffard, Emmanuel Parisien Bourgeois, Victor Desrosiers, Alexandre Langlois, Cédric Simard

Présenté au FI3E par le Cégep de Matane

FORUM INNOVATION
INGÉNIERIE | INFORMATIQUE |
ENTREPRENEURIAT | UQAR

Argus

Service de vision intelligent pour les jeux

Argus donne la vue à vos jeux. C'est une plateforme de vision auto-hébergée prête à brancher à un mini-jeu, un site web ou une app mobile. La démonstration met en scène un personnage qui saute au-dessus d'une voiture ou d'une moto, mais pas d'un avion. **Argus reçoit l'image et renvoie la réponse via une API**. Là où un studio devait jusqu'ici louer une API coûteuse ou monter toute une infrastructure d'IA, Argus offre un service prêt à brancher en quelques lignes de code sans dépendance à un vendeur externe. L'indépendance numérique en intelligence artificielle est rendue accessible aux petits projets.

Pourquoi ? La problématique

Besoin croissant de vision

Beaucoup d'applications modernes auraient besoin de voir : un jeu qui réagit aux obstacles, une app d'accessibilité pour non-voyants, des jeux de réalité augmentée qui voit des vraies cartes sur la table, etc.

Complexité de mise en œuvre

Installer un modèle de détection fiable en production reste un projet à part entière, hors du cœur de métier d'un développeur de jeu. C'est aussi différent du travail du développeur AI qui travaille dans un notebook.

Absence de solution "Prêt à l'emploi"

L'industrie du jeu manque de services autonomes rapides à brancher. Seules les APIs de vendeurs externes sont disponibles (API OpenAI Vision, API AWS Rekognition, API Google Cloud Vision, API Claude Vision, etc.)

Le coût de voir : Panorama des APIs de vision

LES 5 « GROS » (STANDARDS)

Amazon Rekognition
~1k–4k$ / million d'images

Google Cloud Vision
Champion de l'OCR

Microsoft Azure AI
OCR multi-langues

Clarifai
Spécialiste indépendant

IBM Watson (Legacy)
Discontinué en 2021

L'ÈRE DES LLM-VISION

OpenAI Vision
GPT-4o, GPT-4V

Google Gemini Vision
Gemini 2.5 / 3.0

Anthropic Claude
Capacités visuelles Claude 3

Tendance actuelle : Compréhension contextuelle vs simple détection.

SPÉCIALISÉS & ALTERNATIFS

Ultralytics HUB
Le cloud YOLO (Payant)

Roboflow
Vision d'entreprise

Imagga
~500$ / million (Bas coût)

Hive AI
Modération de contenu

Apple Vision
On-device (iOS uniquement)

Argus se positionne comme une alternative autonome pour éviter ces dépendances et coûts récurrents.

L'illusion du "Pay-as-you-go" : Le choc du volume

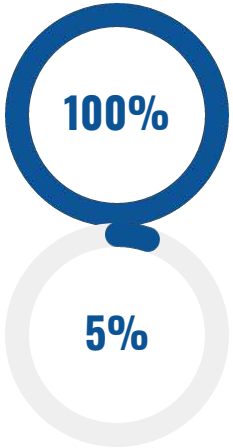
SCÉNARIO : 1 JEU, 1000 JOUEURS

Imaginons un jeu mobile qui analyse 1 image toutes les 3 secondes pour adapter l'environnement au joueur.

- Session moyenne : 15 min (300 requêtes)
- Joueurs actifs : 1000 / jour
- Volume mensuel : **9 millions d'images**

À **1.00$** / **1k** requêtes (Standard AWS/Google), la facture s'élève à **9 000 $ / mois**.

DÉCOMPTE DES COÛTS ANNUELS



API CLOUD (STANDARD)
108 000 $

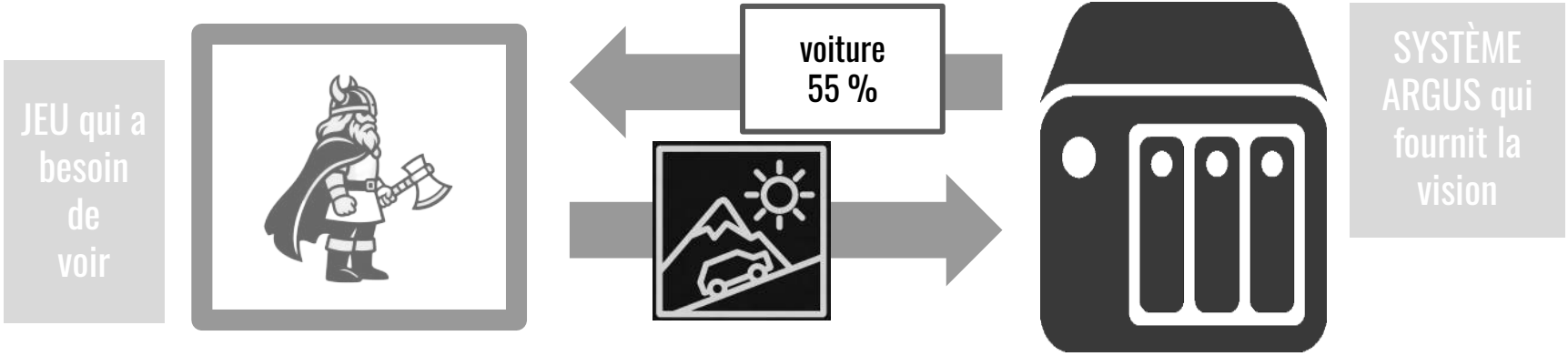
INFRASTRUCTURE AUTONOME
~5 400 $

Le coût d'un serveur dédié est fixe, peu importe le volume.

L'indépendance numérique n'est pas seulement éthique, elle est vitale pour la rentabilité des projets.

Objectifs

Ce projet vise à développer un système de vision intelligente qui reçoit une image par une simple requête web et renvoie son identification, sous forme d'un service modulaire et réutilisable sans sacrifier l'**indépendance numérique** du projet.



L'architecture du sytème doit être préparée avec une emphase particulière sur la **réutilisabilité** et la facilité de déploiement de type clé-en-main.

Méthodologie

1) Infrastructure

Activation du VPS linux sécurisé avec pare-feu et fail2ban.
Mise en place d'un serveur web en reverse proxy authentifié pour exposer un point d'entrée unique sécurisé.

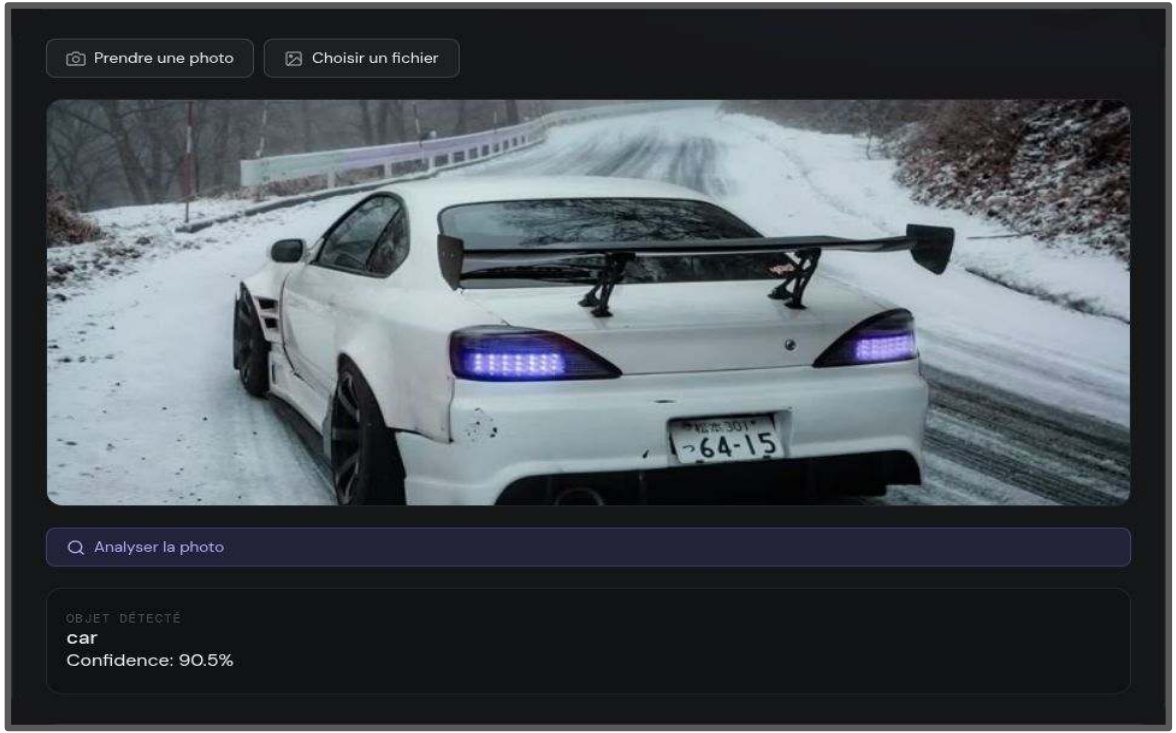
2) Intelligence Artificielle

Installation de serveur CodeProject.AI et sélection du module YOLOv5. Désactivation des autres modules. Optimisation d'inférence et validation de la pipeline avec des images de test.

3) Application Réelle

Développement d'un mini-jeu Unity qui capture le point de vue du personnage et lit la réponse JSON. Préparation du mode temps réel de flux vidéo continu.

Méthodologie - Interface web de test



Vous pouvez tester vous-même à

https://aiimageproject.online/

Catégories permises :
- voitures et motos
- avions et hélicoptères

Méthodologie - Mini-jeu de validation

Développement Unity

Appel à l'API Argus et traitement de la réponse JSON pour l'interaction en jeu.

Logique de Décision

Saut par-dessus les voitures détectées; reste immobile si un avion est identifié.

Mode Temps Réel

Optimisation pour un flux vidéo continu garantissant une validation fluide.

Flux pour un client (exemple du mini-jeu)

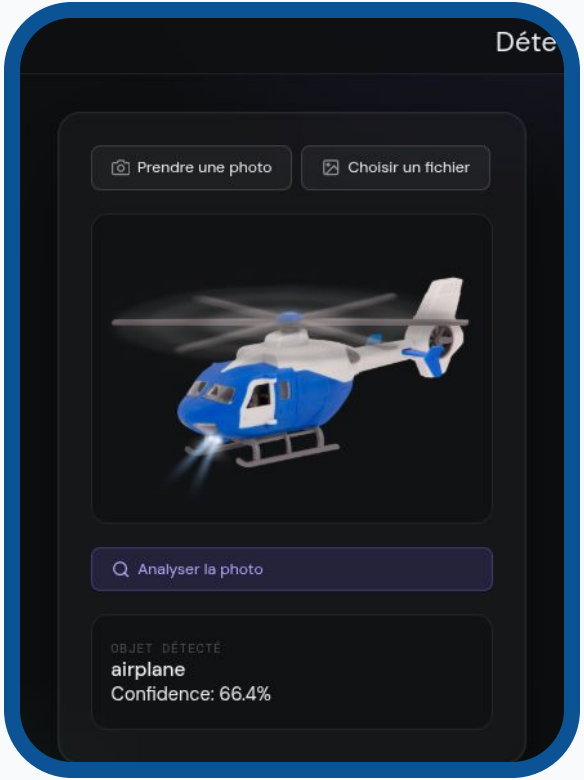
01 1. Le client capture une image

02 2. Requête multipart `POST /v1/vision/detection`

03 3. Nginx authentifie et transmet

04 4. YOLOv5 analyse et renvoie `{ label, confidence, box }`

05 5. Le client applique sa propre logique selon l'information



Matériel et logiciel

Pour répéter l'expérience, vous aurez besoin des éléments suivants :

- **Serveur** :
  Ubuntu 24.04 LTS, 4 Go RAM
- **Sécurité** :
  passerelle Nginx + HTPasswd
- **IA** :
  CodeProject.AI 2.9.5 + YOLOv5

Logiciel : CodeProject.AI-Server

C'est un logiciel open source sur Github qui facilite l'utilisation de l'intelligence artificielle sur son propre serveur.



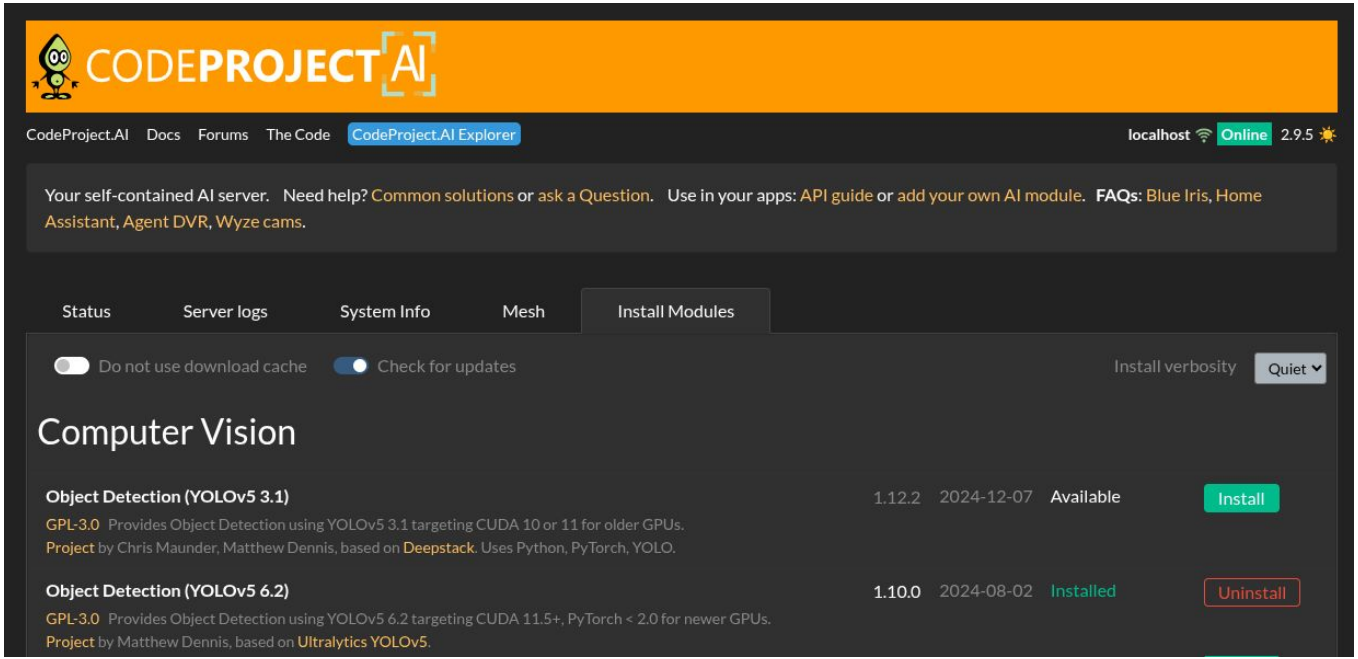


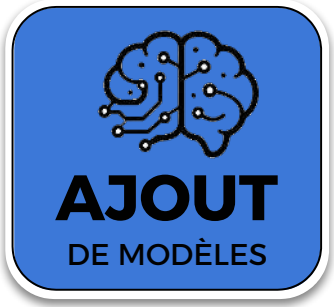

Il permet d'utiliser les possibilités de l'IA pour la détection, l'analyse, la modification de toutes sortes d'images et plus encore.

Les avantages d'utiliser ce service seraient la confidentialité des données, la flexibilité de l'offre et l'autonomie des ressources informatiques.

Logiciel : CodeProject.AI-Server

Voici l'interface d'administration de CodeProject.AI





Résultats : Performance & Sécurité

Workflow & Intégration

- Chaîne : Détection → Score → Décision → Archivage
- 4 sources enrichies en parallèle
- 11 nœuds actifs dans n8n

Leurre OpenCanary

Capture réelle de tentatives SSH :

- Mots de passe : `qwerty`, `1234`
- Ex : IP Tor **185.220.101.1**
- Score 40 (Niveau Moyen) → Firestore

Infrastructures & DevOps

- Script `install.sh` (153 lignes)
- HTTPS automatique & Proxy Nginx
- Interfaces : public vs admin

Validation & Déploiement

100%
Tests Webhooks Validés

153
Lignes Script Install

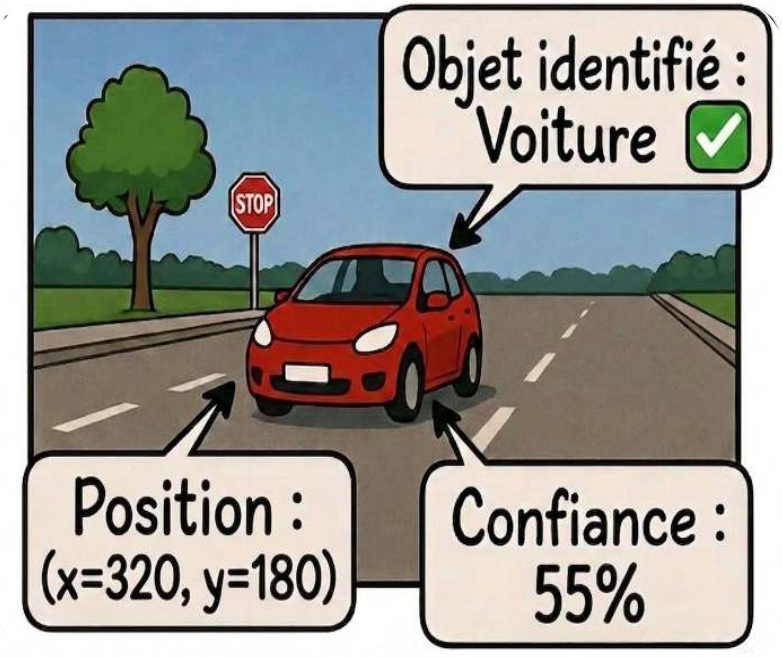
n8n
Workflow Engine

Résultats : Service Industrialisé

Le service **industrialisé*** permet d'obtenir pour n'importe quelle image ses informations signalétiques :

- Objet identifié
- Position précise
- Score de confiance

*Le service fournit un contrat d'API stable encapsulant des modèles versionnés. La latence est contrôlée et le système est résistant aux attaques.



Résultats : on utilise la SENSIBILITÉ (recall)

Sensibilité (aussi appelée rappel ou recall) : répond à la question « sur toutes les vraies voitures qu'on a testées, combien le système a-t-il effectivement reconnues ? »

Une sensibilité de 100 % signifie qu'aucune voiture n'a été oubliée.

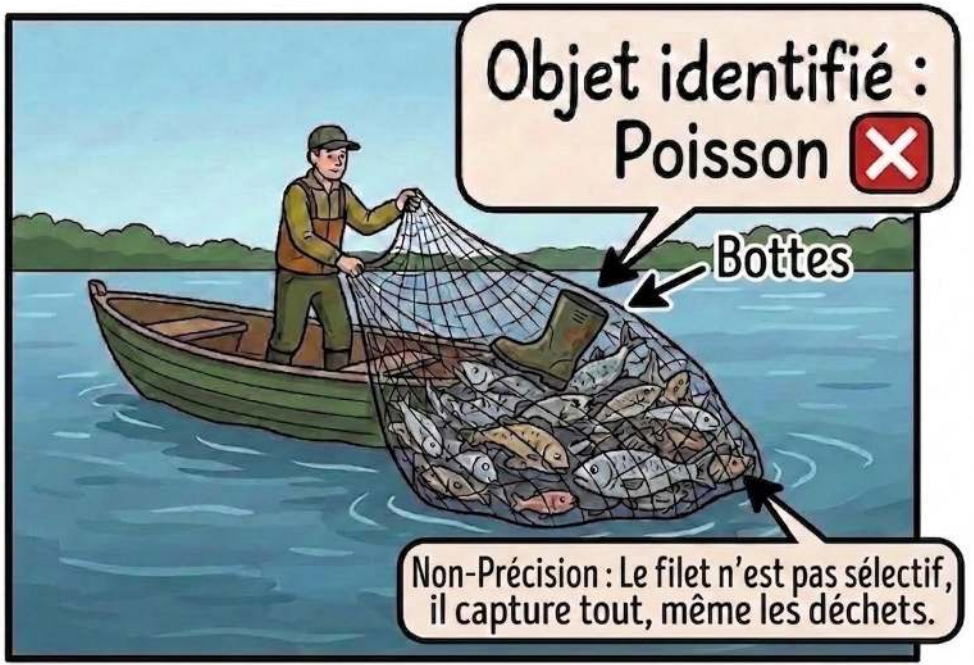
Analogie : un pêcheur sensible attrape tous les poissons qui passent dans son filet.

Résultats : on utilise la PRÉCISION

Précision (ou sélectivité): répond à la question « quand le système dit c'est une voiture, est-ce qu'il a toujours raison ? »

Une précision de 100 % signifie qu'il ne s'est jamais trompé en annonçant «voiture ».
Une sensibilité de 100 % signifie aussi qu'aucune voiture n'a été oubliée.

Analogie : un pêcheur précis ne ramène jamais de bottes dans son filet : tout ce qu'il remonte est vraiment un poisson



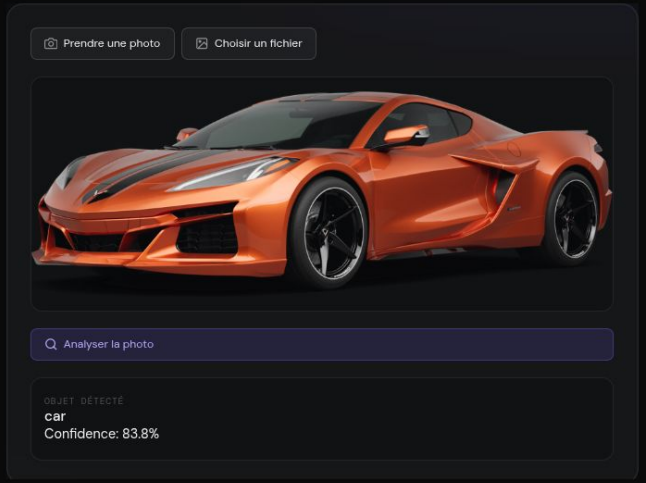
Tous les objets, y compris les bottes, sont pêchés par le filet

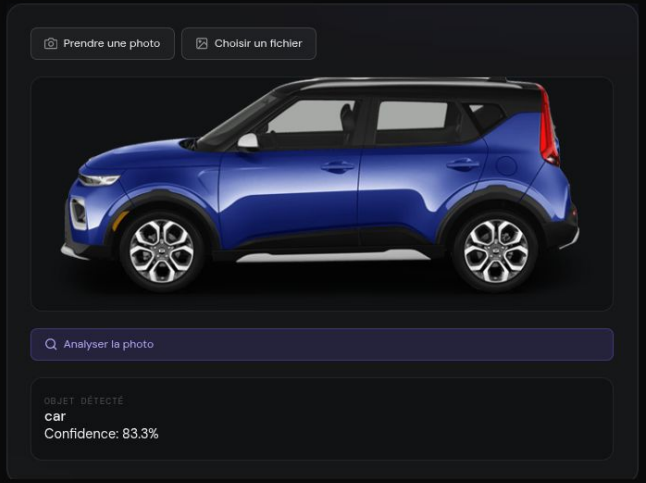
Résultats :

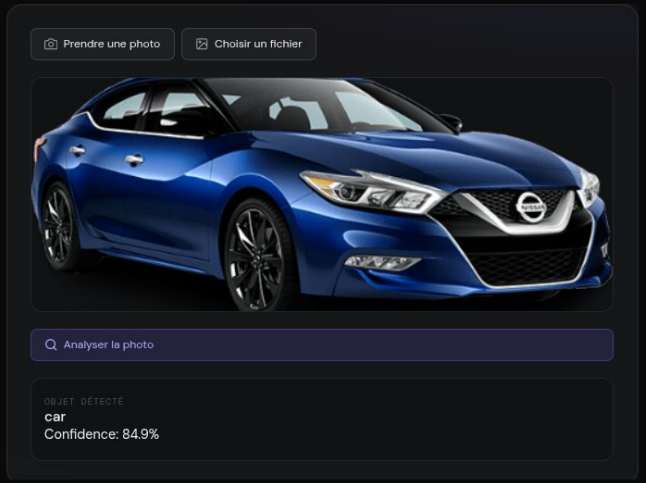
Argus reconnaît les voitures, peu importe le modèle.

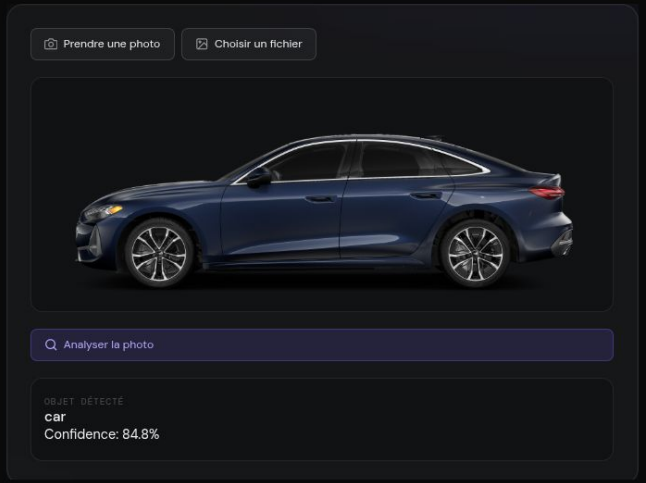
Quatre silhouettes différentes (sportive, familiale, berline, citadine) toutes identifiées voiture avec une confiance entre **83 % et 85 %**.

=> Dans le mini-jeu, cette détection déclenche le saut du personnage.









Performance : était-ce une moto oui ou non ?

Matrice de Confusion (n=21)

	PRÉDIT : CAR	PRÉDIT : AUTRE
RÉEL : CAR	**5 (VP)**	0 (FN)
RÉEL : NON-CAR	0 (FP)	**16 (VN)**

Interprétation : Argus identifie 100% des voitures sans fausse alerte ni aucun oubli sur cet échantillon.

Métriques de Performance

Rappel (Sensibilité)	VP / (VP + FN)	**100 %**
Précision	VP / (VP + FP)	**100 %**
Spécificité	VN / (VN + FP)	**100 %**
Exactitude	(VP+VN) / Total	**100 %**
F1-Score		**1.00**

- Argus a identifié toutes les voitures du jeu de test : aucun oubli => rappel (recall) parfait.
- Argus n'a jamais pris une non-voiture pour une voiture : aucune fausse alarme => précision parfaite.

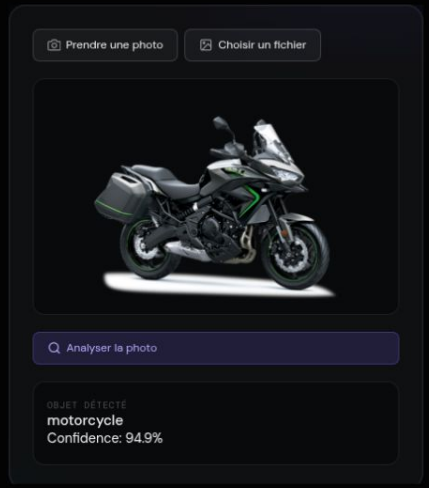
Note : Résultats obtenus en conditions contrôlées sur un petit échantillon. À confirmer sur un corpus plus large.

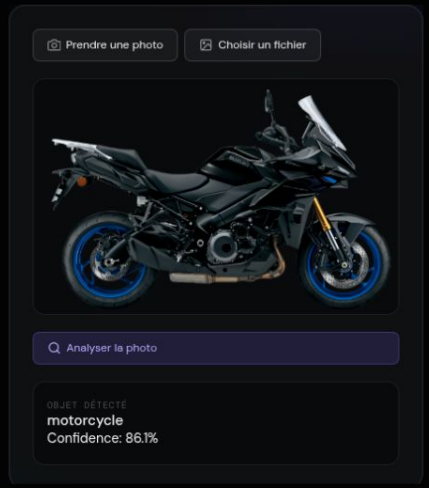
Résultats : détection des motos

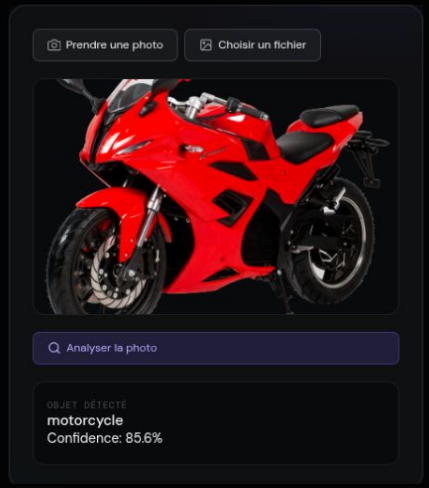
Argus reconnaît les motos, peu importe le modèle.

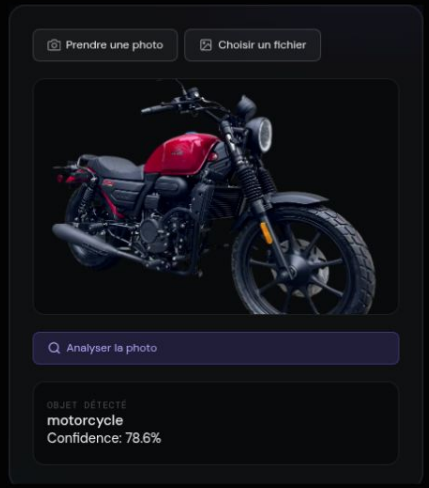
Qu'il s'agisse d'une routière, d'une sportive ou d'une moto classique, Argus renvoie la même réponse : motorcycle, avec une confiance entre 70% et 95 % sur la plus nette. Kawasaki à 94,9 % = record du projet.

=> Dans le mini-jeu, cette détection déclenche le saut du personnage.









Performance : était-ce une moto oui ou non ?

Matrice de Confusion (n=21)

	PRÉDIT : MOTO	PRÉDIT : AUTRE
RÉEL : MOTO	**5 (VP)**	0 (FN)
RÉEL : NON-MOTO	0 (FP)	**16 (VN)**

Interprétation : Argus identifie 100% des motos sans fausse alerte ni aucun oubli sur cet échantillon. La plage de confiance est plus large que pour les voitures.

Métriques de Performance

Rappel (Sensibilité)	VP / (VP + FN)	**100 %**
Précision	VP / (VP + FP)	**100 %**
Spécificité	VN / (VN + FP)	**100 %**
Exactitude	(VP+VN) / Total	**100 %**
F1-Score		**1.00**

- Argus a identifié toutes les motos du jeu de test : aucun oubli => rappel (recall) parfait.
- Argus n'a jamais pris une non-moto pour une moto : aucune fausse alarme => précision parfaite.

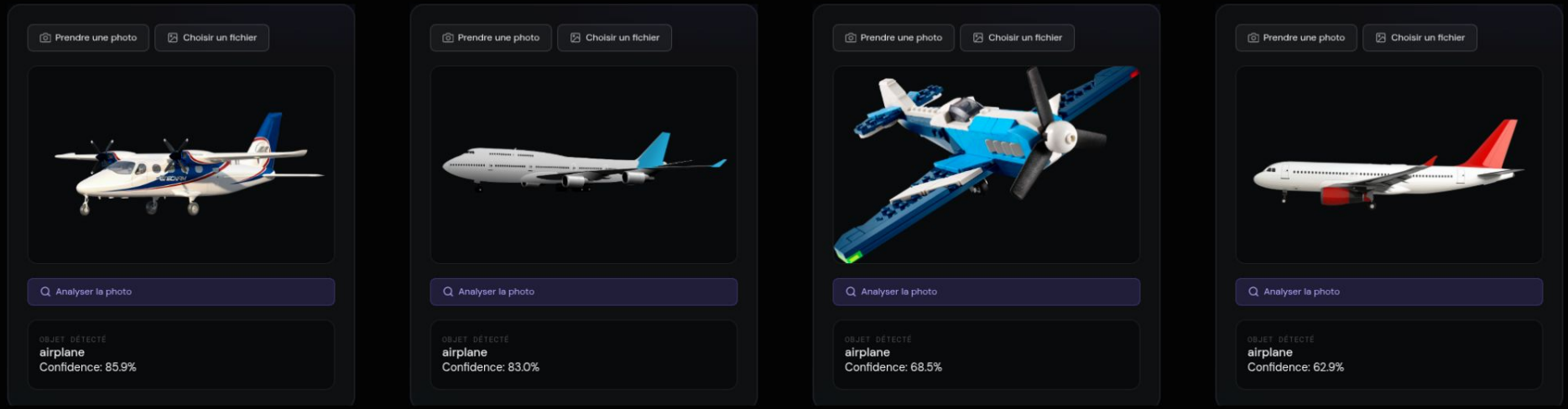
Note : Résultats obtenus en conditions contrôlées sur un petit échantillon. À confirmer sur un corpus plus large.

Résultats : détection des avions

Argus reconnaît les avions, peu importe le modèle.

Même un modèle Lego est reconnu comme avion. jumbo 747, avion de ligne A320, bimoteur privé, jouet de construction : argus reconnait 'airplane' à chaque fois, entre 63 % et 86 % de confiance.

=> Dans le mini-jeu, cette détection déclenche l'immobilité du personnage.



Performance : était-ce un avion oui ou non ?

Matrice de Confusion (n=21)

	PRÉDIT : PLANE	PRÉDIT : AUTRE
RÉEL : PLANE	**4 (VP)**	2 (FN)
RÉEL : NON-PLANE	2 (FP)	**13 (VN)**

Interprétation : Argus identifie 67% des avions sans fausse alerte ni aucun oubli sur cet échantillon. La plage de confiance est la plus large.

Métriques de Performance

Rappel (Sensibilité)	VP / (VP + FN = 4 / (4 + 2)	**66.7 %**
Précision	VP / (VP + FP) = 4 / (4 + 2)	**66.7 %**
Spécificité	VN / (VN + FP = 13 / (13 + 2)	**86.7 %**
Exactitude	(VP+VN) / Total = 17 / 21	**81.0 %**
F1-Score		**66.7 %**

- 2 faux négatifs : les deux avions en dessin/cartoon (plane_3 et plane_4) non détectés
- 2 faux positifs : deux hélicoptères classés comme airplane (helicopter_3 et helicopter_5)

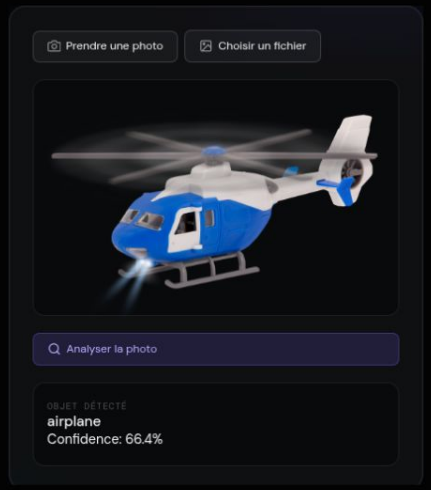
Note : Résultats obtenus en conditions contrôlées sur un petit échantillon. À confirmer sur un corpus plus large.

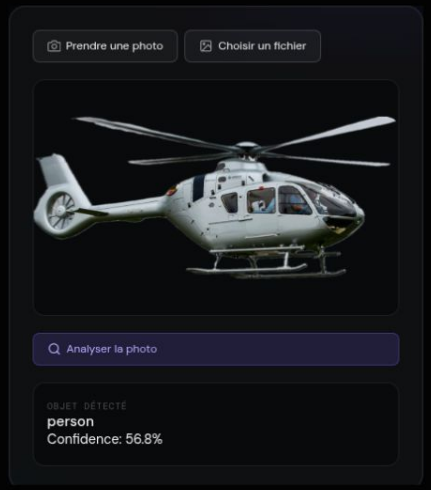
Résultats : détection des hélicoptères

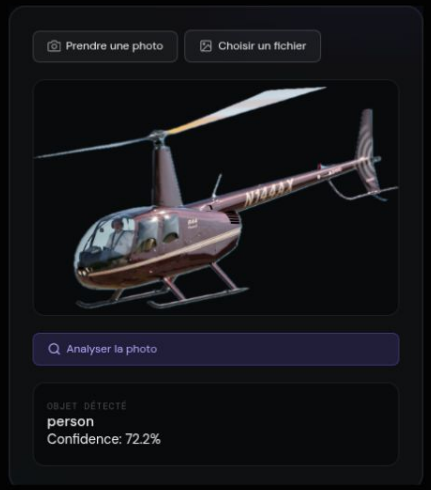
Argus reconnaît les engins volants ou les pilotes !

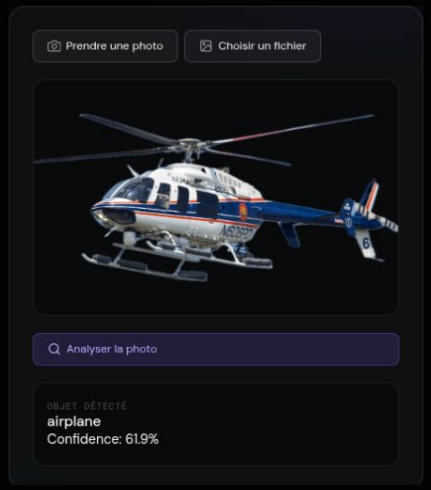
Argus ne connaît pas la classe helicopter : elle n'existe pas dans son modèle. Il voit pourtant toujours quelque chose : l'engin volant (airplane) ou le pilote dans la cabine (person).

=> Dans les deux cas, le jeu prend la bonne décision : ne pas sauter.







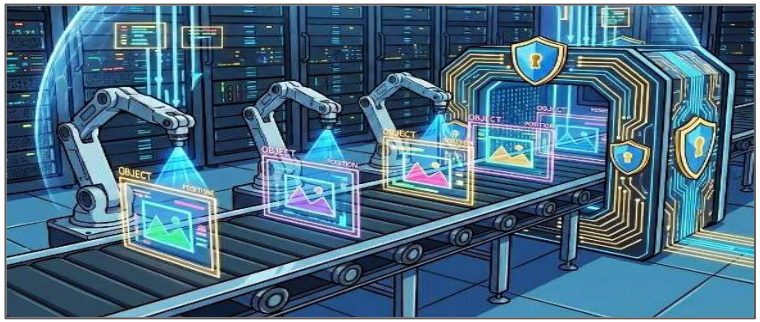


Futur

Les améliorations futures du projet sont de cinq ordres :

Optimisations du Service

- Automatisation du déploiement clé-en-main
- Optimisation et sécurisation continue de la stack technique



Expérience utilisateur & Interface

- Portail libre-service d'inscription pour recevoir une clé d'API et visualiser un tableau de bord
- Application aux caméras de surveillance
- Isolation automatique des images

→ Roadmap évolutive du projet

Futur : Optimisation de la stack technique

Chaque élément de la stack technique peut nécessiter des améliorations fonctionnelles, des sécurisations, des optimisations et une veille technologique.

Serveur

Ubuntu 24.04 LTS,
4 Go RAM

Sécurité

passerelle Nginx
+ HTPasswd

IA

CodeProject.AI 2.9.5
+ YOLOv5 6.2

Pourquoi ARGUS ? Le nom Argus

Argus Panoptès, dans la mythologie grecque, était un géant aux cent yeux, gardien sans sommeil.

Ses yeux étaient répartis sur tout son corps : peu importe où se tournait son visage, il voyait toujours.

Un **Argus moderne** est exactement ce que nous offrons : des yeux prêts à servir toutes vos applications, simultanément, sans relâche.

Références : outils utilisés

CodeProject.AI-Server - https://www.codeproject.com/AI/docs/

YOLOv5 (Ultralytics) - https://github.com/ultralytics/yolov5

Nginx - https://nginx.org/en/docs/

Linode / Akamai - https://www.linode.com/docs/guides/

Ubuntu 24.04 LTS - https://ubuntu.com/server/docs

RFC 7578 - multipart/form-data (pour la requête d'envoi d'image)

RFC 7235 - authentification HTTP (pour HTPasswd)

Références : docs techniques

Documentation des modules CodeProject.AI
https://www.codeproject.com/AI/docs/devguide/module_examples/ObjectDetection.html

Ultralytics : YOLOv5 Documentation
https://docs.ultralytics.com/yolov5/
(Jocher et al., créateurs et mainteneurs du modèle)

Nginx : Reverse Proxy Guide
https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

À propos des algorithmes de LLM et jeux :

Redmon et al., 2016 - You Only Look Once: Unified, Real-Time Object Detection, arXiv:1506.02640 (papier fondateur de YOLO)

Jocher et al., Ultralytics - YOLOv5 Documentation, https://docs.ultralytics.com/yolov5/ (créateurs et mainteneurs du modèle)

Yu et al., Improving Deep Object Detection Algorithms for Game Scenes, Electronics 10(20):2527, MDPI (2021) - application au contexte jeu

Pour en savoir plus sur les LLM et les jeux

Gallotta, R., Todd, G., Zammit, M., Earle, S., Liapis, A., Togelius, J., et Yannakakis, G. N. (2024). Large language models and games: A survey and roadmap. IEEE Transactions on Games. https://doi.org/10.1109/TG.2024.3461510

Hu, S., Huang, T., Liu, G., Kompella, R. R., Ilhan, F., Tekin, S. F., Xu, Y., Yahn, Z., et Liu, L. (2024). A survey on large language model-based game agents (arXiv:2404.02039) [prépublication]. arXiv. https://doi.org/10.48550/arXiv.2404.02039

Hyde-Smith, P. K. (2023). Using object detection to navigate a game playfield [Mémoire de maîtrise, Marquette University]. e-Publications@Marquette. https://epublications.marquette.edu/theses_open/737/

Jung, M., Yang, H., et Min, K. (2021). Improving deep object detection algorithms for game scenes. Electronics, 10(20), 2527. https://doi.org/10.3390/electronics10202527

MERCI