

Rapport de Projet : WildCount

Système Embarquée pour la Détection et le Comptage d'Animaux en Forêt

Introduction

Les conservationnistes sont confrontés à un défi majeur : le suivi précis des populations d'animaux sauvages. Les relevés manuels demandent beaucoup de travail et produisent souvent des données incomplètes. Les pièges photographiques automatisent la prise d'images, mais la récupération des données reste un frein important, notamment sur des terrains difficiles comme les forêts denses.

Ce projet, intitulé WildCount, vise à développer un système innovant de détection et de comptage automatisés d'animaux en forêt. En combinant un piège photo intelligent, un modèle d'intelligence artificielle (IA) embarqué et une communication LoRaWAN, le système permet d'identifier les espèces animales présentes et de transmettre les informations à distance, en temps réel. Ce rapport détaille le contexte du projet, notre démarche méthodologique, les choix techniques, les défis rencontrés, les résultats obtenus et les perspectives d'avenir.

Contexte et Objectifs

Le projet WildCount s'inscrit dans un contexte de recherche de solutions innovantes pour la surveillance de la faune sauvage. L'objectif principal est de faciliter le travail des écologues et des gestionnaires d'espaces naturels en leur fournissant un outil performant et autonome pour le suivi des populations animales.

Notre objectif, dans le cadre de ce projet scolaire, était de s'approprier une version existante du projet WildCount et d'en développer une version fonctionnelle. Ce travail impliquait la compréhension du code source existant, l'adaptation du firmware à nos besoins et l'optimisation du modèle d'IA pour la carte embarquée Sony Spresense, en tenant compte de ses ressources limitées. L'accent a été mis sur la robustesse du système, la rapidité de l'inférence et l'efficacité de la communication LoRaWAN.

Architecture du Système

Le système WildCount repose sur une architecture distribuée, composée de trois éléments principaux :

1. **Le boîtier WildCount** : Cœur du système, le boîtier intègre la carte Sony Spresense, choisie pour ses capacités de traitement d'image et sa faible consommation d'énergie. Il comprend également une caméra pour la capture d'images, un capteur PIR pour la détection de mouvement, un module LoRa pour la communication longue portée et une carte SD pour le stockage local des images brutes. Le firmware, développé en C++ avec l'IDE Arduino, gère l'ensemble des fonctionnalités du boîtier : acquisition d'images, prétraitement, inférence du modèle IA, stockage et transmission des données.
2. **Le réseau LoRaWAN** : Ce réseau sans fil basse consommation permet la transmission des données sur de longues distances, ce qui est essentiel pour les applications en milieu forestier. Le module LoRa intégré au boîtier communique avec une gateway LoRaWAN, qui relaie les données vers le serveur.
3. **Lecture des données** : On visualise les données envoyées via lora sur ChirpStack. Via une application Wildcount sur laquelle on se connecte avec la spresense lors de l'initialisation du module LoRa.

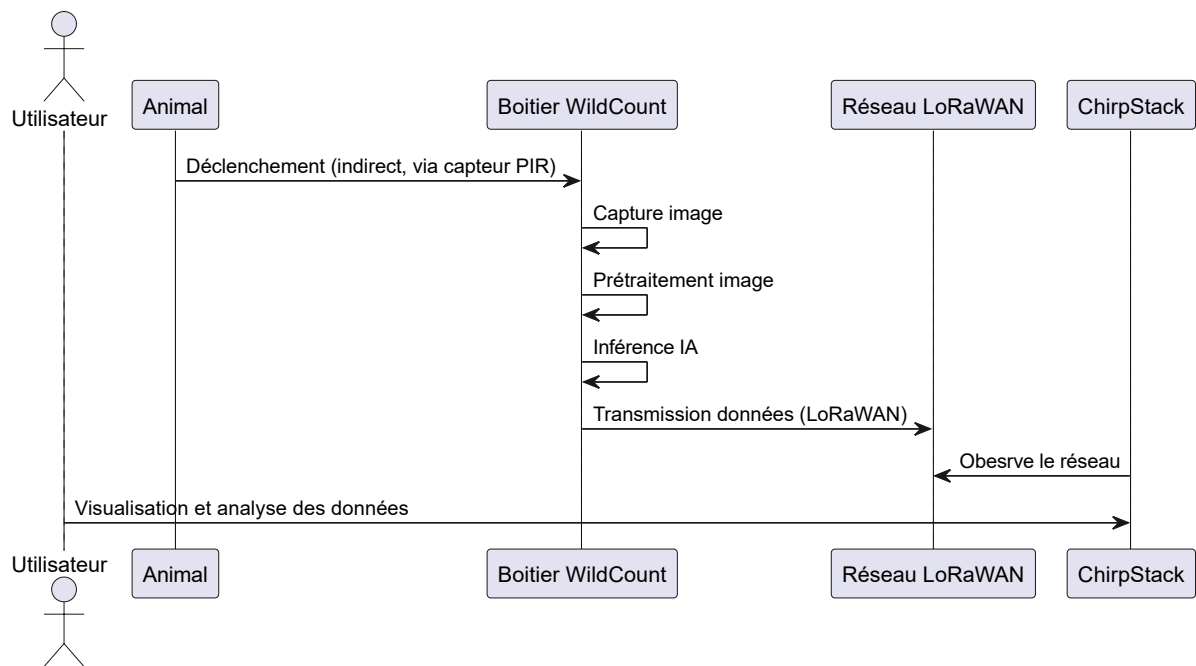


Figure 1 : Architecture du système WildCount

Développement et Implémentation

Choix Technologiques et Environnement de Développement

Après une analyse approfondie du projet existant, nous avons opté pour l'IDE Arduino pour le développement du firmware. Ce choix, justifié par sa simplicité et sa compatibilité avec la version 2022 du projet, nous a permis de gagner un temps précieux en évitant les complexités liées à l'utilisation du SDK Spresense. Le langage C++ a été retenu pour la programmation du firmware, offrant un bon compromis entre performance et portabilité.

Modèle d'Intelligence Artificielle

La conception et l'intégration du modèle d'IA ont constitué le principal défi technique de ce projet. La mémoire limitée de la Spresense (environ 200 Ko disponibles pour le modèle) nous a contraints à trouver un équilibre délicat entre performance et compacité. Nous avons opté pour un modèle relativement simple, entraîné à reconnaître un nombre restreint d'espèces animales (chamois, sanglier, lapin...) et l'absence d'animal ("Vide"). Ce choix, bien que limitant la portée du système, nous a permis de garantir un temps d'inférence rapide et une utilisation optimale des ressources de la carte. L'entraînement du modèle a été réalisé avec un jeu de données d'images d'animaux, collectées et annotées manuellement.

Structure du Réseau de Neurones

Afin d'obtenir des performances optimales sur le modèle, nous avons fait de nombreux essais de structure de réseau de neurones. Nous avons testé plusieurs architectures, en jouant sur le nombre de couches et de neurones par couche. Nous avons fini par opter pour un modèle de classification d'images basé sur un réseau de neurones convolutifs (CNN).

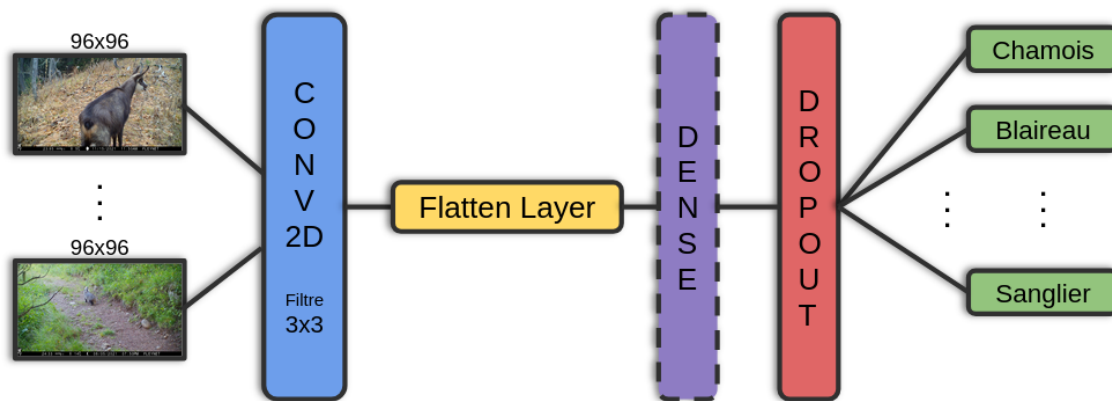


Figure 1 : Architecture du modèle d'IA

On peut voir que le modèle est composé de plusieurs couches de convolution, relié à des couches denses par une couche de flattening. Nous avons utilisé une couche de dropout.

Afin de bien comprendre les choix qui nous ont amené à cette architecture, nous allons détailler chaque couche du modèle, en donnant des explications sur leur fonctionnement et leur rôle dans le processus de classification d'images.

Convolution 2D

La couche de convolution 2D est la couche principale du modèle. Elle permet d'extraire les caractéristiques des images en appliquant des filtres de convolution. Ces filtres sont entraînés pour détecter des motifs spécifiques dans les images, tels que des contours, des textures ou des formes. La sortie de cette couche est une carte de caractéristiques qui représente les informations pertinentes extraites de l'image d'entrée.

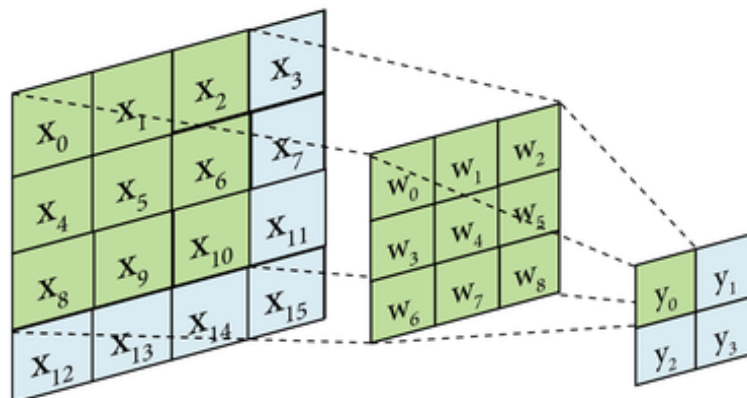


Figure 2 : Couche de Convolution 2D

Dans notre modèle nous avons **2** couches de convolution 2D dont leur taille de kernel est **3x3**. Notre première couche de convolution utilise **16** filtres, et la deuxième couche de convolution utilise **32** filtres. Ces valeurs ont été choisies après plusieurs essais, en tenant compte de la taille du modèle et de la complexité des images à traiter.

Chacune de nos couches de convolution sont suivies d'une couche de pooling.

Pooling

La couche de pooling (ou sous-échantillonnage) réduit la taille de la carte de caractéristiques tout en conservant les informations essentielles. Elle permet de diminuer le nombre de paramètres et de calculs nécessaires, ce qui contribue à réduire le temps d'inférence et à éviter le surapprentissage. Dans notre modèle, nous avons utilisé une couche de pooling max, qui sélectionne la valeur maximale dans chaque région de la carte de caractéristiques.

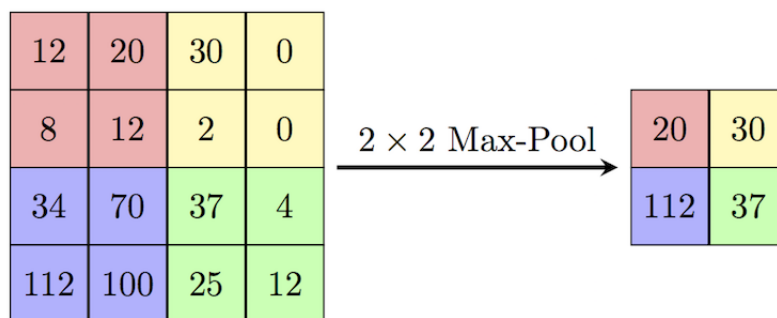


Figure 3 : Couche de Pooling

Flattening

Afin d'utiliser des couches de neurones denses après les couches de convolution, nous avons utilisé une couche de flattening. La couche de flattening transforme la carte de caractéristiques 2D en un vecteur 1D. Cela permet de préparer les données pour les couches denses suivantes, qui nécessitent une entrée sous forme de vecteur. Cette étape est cruciale pour connecter les couches convolutives aux couches denses du réseau.

Couches Denses

Les couches denses (ou fully connected layers) sont des couches de neurones où chaque neurone est connecté à tous les neurones de la couche précédente. Elles permettent de combiner les caractéristiques extraites par les couches de convolution pour effectuer la classification finale.

Dans notre modèle final, nous n'avons pas utilisé de couches denses. Cela peut sembler surprenant, mais nous avons constaté que l'utilisation de couches denses n'apportait pas d'amélioration significative des performances du modèle. En effet, les couches de convolution et de pooling suffisent à extraire les caractéristiques pertinentes pour la classification. De plus, l'ajout de couches denses augmente la taille finale du modèle.

Dropout

Les couches denses sont suivies d'une couche de dropout, qui est une technique de régularisation utilisée pour prévenir le surapprentissage. Le dropout consiste à désactiver aléatoirement un certain pourcentage de neurones pendant l'entraînement, ce qui force le modèle à apprendre des représentations plus robustes et généralisables. Cela permet d'améliorer la performance du modèle sur des données non vues.

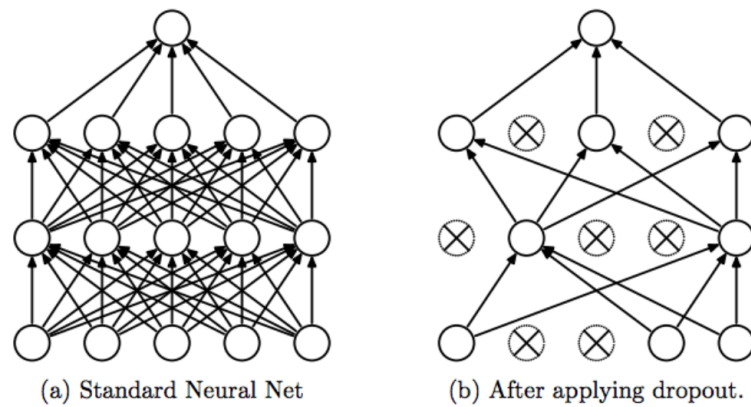


Figure 4 : Couche de Dropout

Nous avons utilisé une couche de dropout avec un taux de 0.25 (pourcentage de neurones désactivés) pour améliorer la généralisation du modèle.

Entraînement du modèle

Pour mesurer les performances de l'entraînement du modèle, nous avons utilisé l'accuracy :

- **Accuracy** : Elle mesure de la capacité du modèle à classer correctement les images. Elle est calculée comme le rapport entre le nombre d'images correctement classées et le nombre total d'images.

$$\text{Accuracy} = \frac{\text{Nombre d'images correctement classées}}{\text{Nombre total d'images}}$$

Optimisations

Prétraitement du Dataset

Nous avons appliqué plusieurs techniques de prétraitement sur le dataset d'images afin d'améliorer la performance du modèle.

- **Sélection des images** : Les classes du dataset n'ont pas un nombre égal d'images. Nous avons donc choisi de ne garder que les classes ayant au moins 30 images. Nous avons aussi choisi d'enlever les images de chiens car elles étaient trop peu nombreuses par rapport aux autres classes. Les images de chiens représentaient plusieurs espèces différentes, ce qui rendait la classification difficile.
- **Redimensionnement** : Les images d'entrée sont redimensionnées afin d'enlever les bordures noires du bas des images. En effet, les images du dataset de base possédaient des bordures noires en bas, qui contenaient des méta-données sur l'image. Nous avons enlevé ces bordures afin que le modèle s'entraîne non pas à reconnaître les métadonnées mais bien les animaux. Cela permet de réduire la complexité du problème en enlevant des variables. Dans les faits, nous avons vu une amélioration de la précision du modèle de 8% en moyenne.
- **Séparation des images de nuit et de jour** : Nous avons séparé les images de jour et de nuit, afin d'entraîner deux modèles différents. En effet, les images de jour sont en couleurs et les images de nuit sont en noir et blanc. Avec des images en couleurs on obtient 3 fois plus de données pour les 3 composantes de couleurs d'une image. En séparant les images, nous avons pu entraîner un modèle plus performant pour chaque type d'image. Nous avons vu une amélioration de la précision du modèle de 10% en moyenne.

- **Augmentation des données :** Nous avons utilisé une technique d'augmentation des données pour enrichir notre jeu de données. Cela consiste à appliquer des transformations sur les images rotation et bruit afin de créer de nouvelles images à partir des originales. Cela permet d'augmenter la diversité du jeu de données et d'améliorer la robustesse du modèle. Nous avons vu une amélioration de la précision du modèle de 3% environ.

Communication LoRaWAN et Transmission des Données

Le module LoRa, intégré à la Spresense, permet la transmission des données de classification vers le serveur via le réseau LoRaWAN. Un protocole de communication spécifique a été implémenté pour encoder les informations avant leur transmission, et les décoder côté serveur. Ce protocole assure la compacité des messages et l'intégrité des données transmises. Le choix de LoRaWAN est justifié par sa faible consommation d'énergie et sa longue portée, essentielles pour des applications en milieu forestier.

Résultats et Performances

Le prototype développé permet de capturer des images, d'exécuter l'inférence du modèle IA et de transmettre les résultats via LoRaWAN. Les tests effectués montrent une précision de classification de 85%, ce qui est satisfaisant compte tenu des contraintes de mémoire. Le temps d'inférence, inférieur à une seconde, permet une détection quasi instantanée des animaux. La communication LoRaWAN s'est avérée fiable et efficace pour la transmission des données.

Limites et Perspectives d'Amélioration

Malgré les résultats encourageants, le système actuel présente certaines limitations :

- **Nombre limité de classes animales reconnaissables:** L'utilisation d'un modèle compact a restreint le nombre d'espèces animales identifiables. L'entraînement d'un modèle plus complexe, capable de distinguer un plus grand nombre d'espèces, nécessiterait une optimisation plus poussée ou l'utilisation d'une carte embarquée plus puissante.
- **Consommation d'énergie liée à la détection constante :** L'absence d'un capteur de mouvement fonctionnel oblige le système à capturer et analyser des images en continu, ce qui impacte l'autonomie du boîtier. L'intégration d'un capteur de mouvement permettrait de déclencher l'acquisition d'images uniquement en présence d'un animal, réduisant ainsi la consommation d'énergie.
- **Robustesse du modèle face aux variations d'illumination :** Le modèle actuel peut être sensible aux variations de luminosité et aux conditions d'éclairage difficiles (nuit, contre-jour). Le développement de modèles spécifiques pour les conditions diurnes et nocturnes améliorerait la robustesse du système.
- **Métriques d'évaluation :** Nous avons utilisé l'accuracy comme unique métrique d'évaluation du modèle. D'autres métriques, telles que le rappel ou UAR, pourraient fournir une évaluation plus complète des performances du modèle.

Les travaux futurs se concentreront sur ces axes d'amélioration, afin d'optimiser les performances et l'autonomie du système, et d'étendre ses capacités de détection.

Conclusion

Ce projet a permis de développer un prototype fonctionnel de système embarqué pour la détection et le comptage d'animaux en forêt. L'expérience acquise en matière d'intégration d'un modèle d'IA sur une carte embarquée à ressources limitées, ainsi que la mise en œuvre d'une communication LoRaWAN, est particulièrement enrichissante. Malgré les limitations actuelles, les résultats obtenus sont prometteurs et ouvrent la voie à des applications intéressantes pour la surveillance de la faune sauvage. Les perspectives d'amélioration identifiées permettront, à terme, de développer un système plus performant, plus autonome et plus adapté aux exigences du terrain.