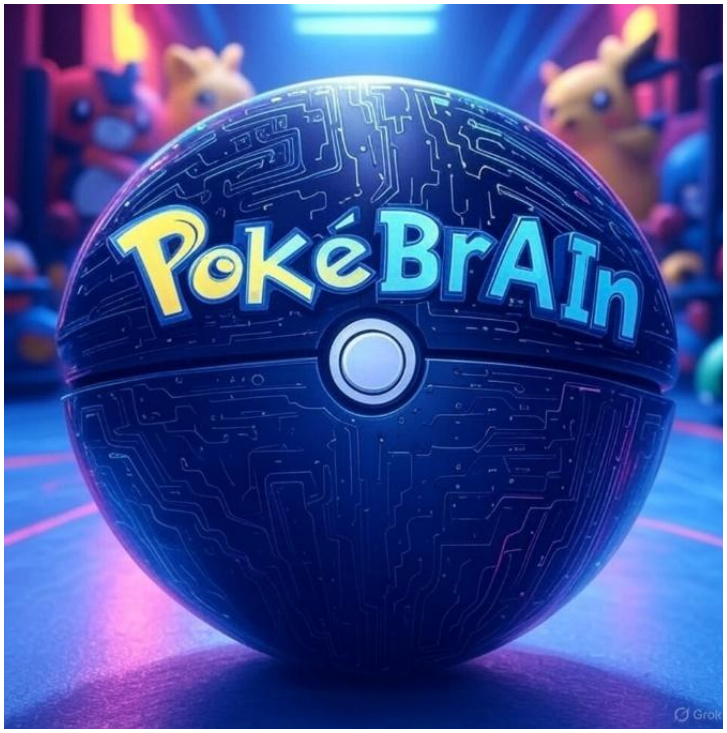
XUEREF Alexis GUÉHÉNEUX Tiago

Projet d'Informatique Robotique



PokéBrAIn,

L'intelligence embarquée au service des cartes rares.

POLYTECH NICE SOPHIA
UNIVERSITÉ CÔTE D'AZUR

Table des matières

I. Introduction

Dans le cadre de notre projet d'informatique de fin de semestre, il nous a été demandé de réaliser une Intelligence Artificielle capable de détecter un son, un geste ou bien une image... Tout ceci en utilisant un modèle pré-entrainé cohérent. Dans un second temps, il nous sera demandé de l'adapter au mieux à notre problématique avant de l'expérimenter dans un cas réel.

Passionné depuis l'enfance par l'univers Pokémon et avec un regain d'intérêt pour les jeux de cartes à collectionner, notamment avec les cartes Pokémon. De nombreux passionnés possèdent des centaines, voire des milliers de cartes. Pourtant, leur gestion reste souvent manuelle, fastidieuse, et sujette à l'erreur. D'autant plus que certaines cartes aujourd'hui sont des véritables perles rares et potentiellement une mine d'or, l'automatisation de leur enregistrement nous a semblé être une bonne problématique à essayer de répondre.

C'est pourquoi, nous avons décidé pour notre projet de réaliser une reconnaissance de cartes Pokémon sur une Jetson Nano. Plus précisément, ce projet vise à construire un système embarqué capable d'identifier des cartes Pokémon à l'aide de l'intelligence artificielle. Mais pas n'importe comment. L'idée, c'est de construire un système embarqué, autonome, capable d'identifier une carte simplement en la montrant à une caméra, puis de la classer dans un inventaire numérique puis si possible ajouter certaines fonctionnalités comme le mettre dans un classeur type « Pokedex » ou bien d'afficher le prix total d'un paquet.

Pour ce faire, nous devons suivre une succession d'étapes :

- Construire ou compléter un jeu de données contenant différentes cartes Pokémon
- Choisir un modèle de Deep Learning léger, mais suffisamment performant
- Entraîner ce modèle, avec une stratégie adaptée au faible nombre d'images par classe
- Déployer le modèle sur un environnement à ressources limitées (Jetson Nano), en garantissant des performances acceptables
- Créer une interface simple pour l'utilisateur (affichage du résultat, stockage des données, interaction possible via capteurs ou boutons)

En somme, ce projet est à la croisée de plusieurs domaines : la vision par ordinateur, l'intelligence artificielle, du système embarqué, du fine-tuning... Il mobilise des compétences en apprentissage profond, en traitement d'image, en systèmes embarqués, et en conception logicielle. Ce rapport présente notre démarche, les choix techniques réalisés, les résultats obtenus, ainsi que les pistes d'amélioration.

II. Problématique et usage

Nous nous demandons donc alors comment concevoir un système embarqué, autonome et transportable, capable d'identifier automatiquement des cartes Pokémon à partir d'images capturées en temps réel, en fournissant pour chacune des informations clés comme le nom, la rareté et la valeur estimée, tout en respectant les contraintes de performance, de légèreté du modèle et de robustesse face aux conditions d'utilisation réelles comme celles rencontrées en brocante ?

III. Choix technologiques

Choix matériel :

Dans le cadre de notre projet, le choix des composants matériels et logiciels n'a pas été arbitraire. Il a été guidé par les contraintes spécifiques de notre cas d'usage : un système embarqué, autonome, léger mais qui est assez puissant pour pouvoir faire tourner notre modèle d'intelligence artificielle.

Nous avons opté pour la carte NVIDIA Jetson Nano comme unité centrale de traitement. Ce choix repose avant tout sur sa capacité à exécuter localement des modèles de Deep Learning tout en maintenant un encombrement réduit et une consommation modérée. La Jetson Nano nous offre un environnement Linux complet, avec une compatibilité directe avec les bibliothèques que nous utilisons, notamment PyTorch et OpenCV. Elle combine donc puissance de calcul et accessibilité, ce qui en fait une plateforme idéale pour le traitement embarqué de l'image en temps réel.

Pour la capture des images, nous avons retenu la webcam Logitech C505e. Son installation plug-and-play sur la Jetson, sa compatibilité immédiate avec les outils de traitement vidéo, et sa qualité d'image satisfaisante, même dans des conditions d'éclairage imparfaites, en font un choix fiable. La simplicité de sa configuration a également permis de concentrer nos efforts sur le traitement des données, sans passer de temps sur des problèmes matériels ou de compatibilité.

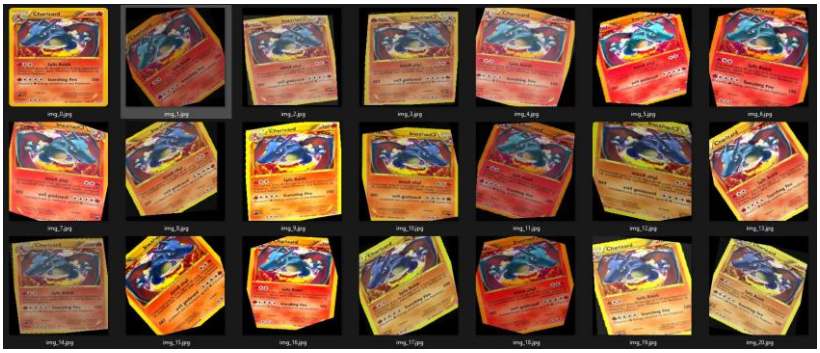
Au début du projet, nous avons réalisé les premiers entraînements de nos modèles sur Google Colab, profitant ainsi de l'accès gratuit à des TPU pour accélérer les phases d'entraînement. Cependant, nous avons rapidement constaté que la puissance et la mémoire disponibles sur ces TPU ne suffisaient pas à entraîner efficacement nos modèles, notamment du fait de la taille et de la complexité des données. Cette limitation a motivé notre choix de poursuivre le développement directement sur une machine locale plus performante, en l'occurrence un ordinateur portable Asus TUF A17. Ce dernier, équipé d'une carte graphique dédiée, nous a permis d'entraîner nos modèles avec plus de flexibilité et de rapidité, tout en conservant la maîtrise complète de l'environnement.

Enfin, afin d'assurer la stabilité et la reproductibilité du système, nous avons décidé de déployer notre application dans un conteneur Docker. Cette approche nous permet de figer l'environnement logiciel, de contrôler les versions des bibliothèques utilisées et de faciliter les tests comme les déploiements. Cela garantit un fonctionnement cohérent sur la Jetson Nano, tout en nous laissant la possibilité de migrer vers d'autres plateformes similaires à l'avenir si nécessaire.

Choix logiciel :

Pour notre projet de reconnaissance de cartes Pokémon, nous avons utilisé le modèle ResNet-18, un réseau de neurones convolutifs léger et efficace. Ce modèle a été pré-entraîné sur ImageNet, puis réentraîné (fine-tuning) sur notre propre dataset.

Nous avons constitué ce dataset en téléchargeant l'intégralité des cartes disponibles via l'API pokemontcg.io, soit près de 18 831 images couvrant toutes les extensions et variantes. Chaque image représente une classe distincte. Pour améliorer la robustesse du modèle face aux conditions réelles, nous avons appliqué des techniques de data augmentation (rotations, changements de luminosité), générant ainsi un dataset encore plus varié d'environ 370 000.

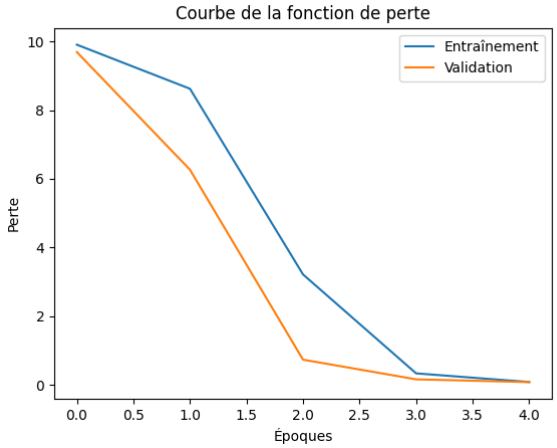


Dracaufeu augmenté par notre IA

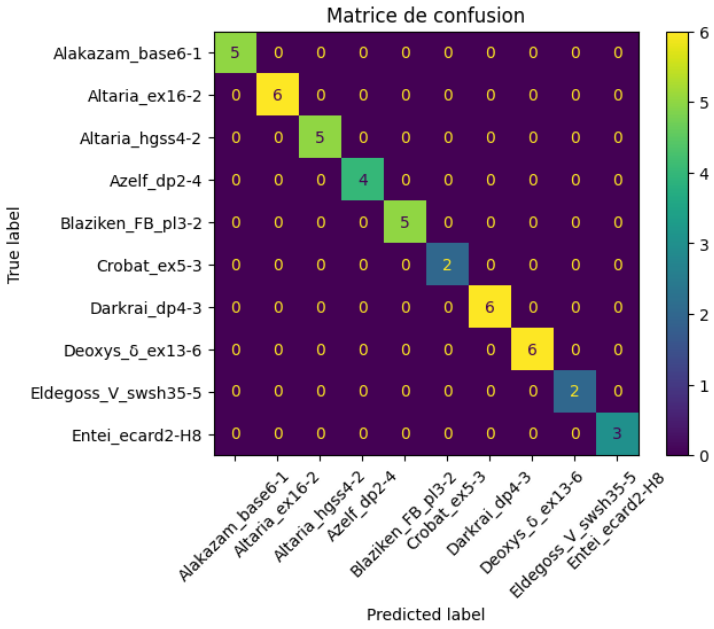
Le modèle a ensuite été converti en TorchScript pour permettre une exécution rapide et optimisée sur notre plateforme embarquée, la Jetson Nano, en temps réel avec une caméra Logitech.

IV. Performances

Après l'entraînement du modèle ResNet-18 sur notre dataset de près de 19 000 cartes Pokémon, enrichi par des techniques de data augmentation, nous avons obtenu une accuracy de validation de 97,96 %. Ce résultat témoigne de la précision élevée du modèle dans la reconnaissance des cartes, même dans des conditions variées de prise de vue.

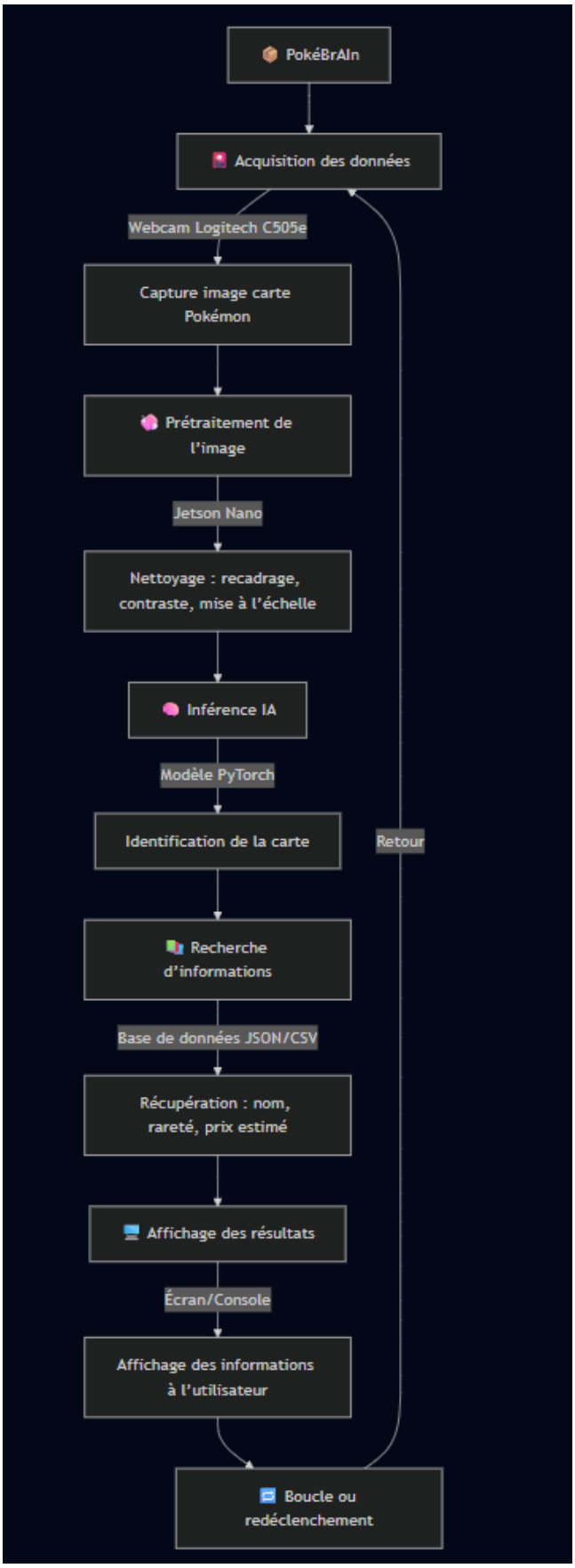


Nous avons aussi effectué des tests de matrice de confusion et avons réalisé un sans-faute.



Ces performances confirment la pertinence du choix du modèle ainsi que la qualité et la diversité du dataset utilisé et amélioré.

Nous avons réalisé un schéma expliquant comment le modèle fonctionne afin de facilement comprendre chaque étape du modèle :



Par la suite, notre modèle renvoie le nom de la carte qu'il a identifié [1] .

Programme reconnaissant une image dans le dossier :

```
(venv) PS C:\Users\a_xue\Desktop\ROB S6\Info S6\Projet_Pokemon\pokemon-card-director> python .\classify_and_inventory.py
✅ Fichier metadata chargé.
📷 Image 'charizard.jpg' classifiée comme : Charizard_cel25c-4_A
🃏 Carte: Charizard_cel25c-4_A | Valeur: 110.0 € | Total: 1
📷 Image 'tepig.jpg' classifiée comme : Tepig_bw1-15
🃏 Carte: Tepig_bw1-15 | Valeur: 0.74 € | Total: 1

=== 💾 Inventaire final ===
Charizard_cel25c-4_A : 1 exemplaire(s), valeur unitaire : 110.0 €
Tepig_bw1-15 : 1 exemplaire(s), valeur unitaire : 0.74 €

💾 Inventaire sauvegardé dans 'outputs/inventaire.csv'
```

Inventaire :

```
1  Nom de carte,Quantité,Valeur unitaire (€),Valeur totale (€)
2  Charizard_cel25c-4_A,1,110.0,110.0
3  Tepig_bw1-15,1,0.74,0.74
```

V. Répartition des taches :

Au lancement du projet, nous avons collaboré à distance via Google Colab, ce qui nous permettait de travailler en temps réel sur le même code, notamment durant les phases d'expérimentation et de prototypage du modèle. Cependant, les limitations en matière de ressources GPU sur cette plateforme se sont rapidement fait sentir, notamment pour l'entraînement du réseau. Cela nous a conduit à adapter notre organisation et à passer sur une machine locale plus performante.

C'est alors Alexis qui a pris en charge l'entraînement, les tests et les ajustements du modèle, grâce à son ordinateur personnel (Asus TUF A17), doté d'une carte graphique suffisamment puissante pour mener à bien ces opérations localement. Le choix du modèle ResNet18, à la fois léger et efficace pour la classification d'images, a également été proposé par Alexis.

Tiago s'est, quant à lui, concentré sur la recherche du dataset utilisé pour l'apprentissage, ainsi que sur l'intégration de la webcam avec le système. Etant donné qu'Alexis se charge de l'entrainement, cela a été normal que la majorité du rapport soit fait par Tiago, ainsi que pour la présentation orale.

Au fil du projet, les tâches se sont donc naturellement réparties en fonction des ressources matérielles disponibles et des compétences spécifiques de chacun. Cependant, le travail s'est partagé de manière régulière par mail et par zip [2] pour tester le modèle chacun de son côté.

VI. Difficultés rencontrées

Au cours de notre projet, nous sommes tombés sur quelques difficultés qui nous ont fait perdre pas mal de temps.

Tout d'abord, nous avons travaillé conjointement sur Google Colab. Cet environnement collaboratif nous permettait d'intervenir simultanément sur le code, en particulier pour les phases d'exploration et de prototypage du modèle. Cependant, après avoir atteint les limites de calcul imposées par la plateforme, notamment en ce qui concerne l'entraînement sur GPU, nous avons dû repenser notre organisation technique.

Mais le plus gros problème rencontré a été la comptabilité liée à Docker sur la Jetson. En effet, nous avons eu plusieurs incompatibilités entre les images Docker et les bibliothèques CUDA, ce qui provoquait des erreurs critiques lors de l'exécution de PyTorch. De plus, nous avons eu pas mal de problèmes lors de la connexion de la caméra, en effet, on avait un problème de versions entre les versions Python et Pythonvision entre celles installées sur la Jetson et celles utilisées dans notre modèle. Ceci nous a pris beaucoup de temps à trouver le problème et à recompiler tout le modèle.

De plus, lors du test de l'IA avec la caméra, nous nous sommes rendu compte que la qualité de la caméra rendait difficile la reconnaissance des cartes a taille réelle, c'est pourquoi nous sommes obligés d'imprimer des cartes plus grandes que les dimensions réelles.

VII. Prochaines étapes du projet :

A la fin de notre projet, nous nous sommes directement posé la question suivante, pouvons-nous en faire une application mobile ?

Cela pourrait marcher comme Google Lens de manière immédiate, il suffirait d'ouvrir son appareil photo puis l'IA analyserait toutes les cartes devant l'appareil avant de les enregistrer, de renvoyer le nom de chaque carte, son édition, son prix. Elle pourrait répertorier ces cartes dans le profil afin d'avoir un historique de toutes les cartes scannées. Cela rendrait le système plus intuitif et accessible au grand public et amateurs de cartes pokémon. On pourrait dans un second temps, s'associer à une application de vente direct de cartes Pokémon afin de rendre l'application tel un marché de cartes.

De plus, on pourrait étendre l'IA à d'autres jeux de cartes comme Magic ou Yu-Gi-Oh ou bien des jeux de cartes de sport comme las célèbres Panini.

Enfin, à la suite de nos tests, on a remarqué que la qualité de la caméra n'était pas bonne et diminuait grandement la bonne détection de la carte. Ainsi, nous pensons à ajouter des images floues et avec des reflets afin de compenser la faible résolution de la caméra. Autre solution serait d'acheter une caméra de meilleure qualité.

VIII. Conclusion :

Le projet PokéBrAIn repose sur une idée simple : reconnaître automatiquement des cartes Pokémon, dans des conditions réelles, à partir d'images prises en direct. L'image, capturée par une webcam, est envoyée vers un modèle d'intelligence artificielle embarqué. Ce modèle, optimisé pour la Jetson Nano, analyse l'image, identifie la carte, puis extrait des informations utiles comme son nom, sa rareté ou sa valeur estimée.

Tout fonctionne localement. Aucun besoin de connexion. Cela rend le système rapide, autonome, et utilisable sur le terrain, par exemple dans une brocante.

Le traitement s'enchaîne sans rupture : capture, prétraitement, prédiction, affichage. L'ensemble reste fluide. Et surtout, il est robuste. Même en conditions imparfaites, cartes inclinées, lumière naturelle changeante, le système fonctionne. Il identifie bien. Il restitue clairement les résultats.

Certes, tout n'est pas parfait. Le modèle reste sensible aux cartes très proches visuellement. Il dépend aussi d'une base embarquée pour les valeurs, ce qui limite la mise à jour automatique des prix. Mais ces limites sont connues. Et elles ouvrent la voie à des améliorations futures.

Parmi elles : développer une application mobile. Cela permettrait d'utiliser le système depuis un simple smartphone. L'utilisateur prendrait la photo avec son appareil. Le traitement pourrait se faire en local, ou via un serveur. Et les informations s'afficheraient en retour. L'interface serait plus directe, plus intuitive.

IX. Annexes :

Annexe 1 - Prédiction à la caméra :

Annexe 2 – Lien GitHub de PokéBrAIn :

https://github.com/AlexisXueref/PokeBrAIn