République Algérienne Démocratique et Populaire

Ministére de l'Enseignement Supérieure et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'informatique

Filiére :

Master Informatique Visuelle

Mémoire de Master

Théme :
Détection et Classification de Météores Lumineux Capturés par des Caméras Allsky

Encadré par :
Dr. Bouyahiaoui Zineddine
Mr Baba Aissa Djounai
Dr. Mebtouche Naoual

Présenté par :
Benkhelifa Abdelghafour
Kaddouri Nassim

Devant le jury composé de :
Dr. Khellaf Faiza
Dr. Dahmane Afifa

Binôme : MIV_07/2024

Remerciement et dédicace

Remerciement

In the name of Allah, the Most Gracious, the Most Merciful.
Alhamdulillah, praise be to Allah the Almighty of God.
We would like to express our gratitude to Allah, who, with His grace, has granted us the strength and wisdom to achieve this significant milestone.
We extend our sincere thanks to the members of the jury. First, for their pivotal roles as our professors throughout our journey as Visual Computing students, and second, for their role in evaluating and adjudicating our work.
We are deeply indebted to our supervisors, Dr. MEBTOUCHE Nawel, Dr. BOUYAHIAOUI Zineddine, and Mr. BABA AISSA Djounai as well as Mr. DAIFFALLAH Khalil. Their help, invaluable support, and guidance have been instrumental in shaping this dissertation. They have not only been mentors but also sources of inspiration throughout this endeavor.
We are also grateful to all those who have supported and encouraged us along the way, including our families, friends, and colleagues.
Thank you all for your contributions and belief in this journey.

Dedicace

In the name of Allah, the Most Gracious, the Most Merciful. Alhamdulillah.
I would like to dedicate this work to my parents, whose unwavering support, sacrifices, and endless love have been the foundation of all my achievements. Your belief in me and constant encouragement have given me the strength and determination to pursue my dreams. Thank you for always being there, guiding me, and shaping me into the man I am today. I am forever grateful for everything you have done.
To my sisters, whose support has been invaluable.
To my friends, who have been my companions during these years. Each one of you, by name, has contributed to this journey.
To every teacher, professor, and person who has taught me, even a single letter – your guidance was a step that led me here.
Thank you all.
"Ambition is the path to success. Persistence is the vehicle you arrive in."
Benkhelifa Abdelghafour

Dedicace

I would like to dedicate this work to my parents and my grandparents. I would have given up a long time ago if not for their support and encouragements. I hope I'll always make them proud of me.

To my friends, who taught me so much and who always pushed me to never stop chasing my dreams.

To my friend and mentor, from whom I learned how to be better as person and as a learner.

To my teacher of undergrad and now master's. I wouldn't be able to be here without her.

And to everyone that has helped shape me into the person I am today.

Thank you all.

"In my experience, there's no such thing as luck." — Obi-Wan Kenobi

Nassim Kaddouri

Table des matières

Table des figures

Liste des tableaux

Résumé

La détection des météores présente un mélange unique de défis et d'opportunités dans le domaine de l'observation astronomique. Ce mémoire s'engage dans un voyage innovant pour améliorer la détection des météores à l'aide de techniques avancées d'apprentissage profond. Nous nous sommes plongés dans les complexités des météoroïdes, des météores et des météorites, en explorant leur signification scientifique et les complexités liées à leur détection.

Dans notre projet, nous avons constitué un solide ensemble de données et expérimenté des modèles de détection d'objets, notamment Faster RCNN avec ResNet et Swin Transformer backbones. Notre approche a impliqué de multiples ajustements d'entraînement et d'évaluation de modèles, chacun affinant la précision de la détection et minimisant les faux positifs.

Dans ce travail nous également développé une interface utilisateur graphique (GUI) intuitive qui permet aux utilisateurs de tester des images et des vidéos pour la détection des météores. Cette interface prend en charge diverses configurations de modèles et introduit une fonction de détection multi-événements, synchronisant plusieurs sources vidéo pour identifier les occurrences simultanées de météores.

Ce mémoire ouvre la voie à de futures avancées dans le domaine de la détection des météores, en proposant une solution pratique et déployable qui intègre la recherche théorique et l'application dans le monde réel. Les résultats démontrent des améliorations significatives dans la précision de la détection, ouvrant la voie à de nouvelles explorations dans ce domaine captivant.

Introduction Générale

Les météorites sont des messagers de l'espace, porteurs d'informations cruciales sur la formation et l'évolution de notre système solaire. Chaque météorite est une capsule temporelle, préservant des indices sur les conditions qui existaient bien avant la naissance de la Terre. En étudiant ces fragments célestes, les scientifiques peuvent reconstituer l'histoire de notre univers. Par exemple, la météorite de Erg Chech, avec ses minéraux anciens, nous offre un aperçu direct des processus qui ont façonné le système solaire primitif. Ces études peuvent révéler des détails sur la formation des planètes, l'origine de l'eau sur Terre, et même les conditions nécessaires à l'émergence de la vie.

Mais, y aurait-il un moyen pour faciliter la recherche de ces objets fascinants ?

De nos jours, une branche de l'astronomie observationnelle s'est dédiée à la détection de météores à l'aide de données vidéos. En effet, il est maintenant possible d'observer et de surveiller le ciel avec des caméras capables de capturer des images couvrant l'ensemble de l'horizon, a la recherche de météores. Ce type de caméras est nommé All Sky qui veut dire littéralement tout ciel. Plusieurs travaux ont utilisé des algorithmes de traitement d'image classiques pour la détection de météores. Bien que ces techniques aient prouvé leur efficacité, elles atteignent leurs limites en termes de précision et de robustesse face à la variabilité des conditions atmosphériques et du bruit. Les méthodes traditionnelles cèdent de plus en plus de place au deep learning. Les réseaux de neurones convolutionnels (CNN), ont révolutionné le domaine de la vision par ordinateur, offrant des performances inégalées pour la détection et la classification d'objets sur des images complexes. Grâce à leur capacité à apprendre des caractéristiques hiérarchiques directement à partir des données brutes, ces modèles surpassent les approches classiques en termes de précision et de résilience. Cependant, l'un des défis principaux de l'utilisation du deep learning pour la détection de météores réside dans l'indisponibilité de données annotées. Pour qu'un modèle de deep learning puisse être entraîné efficacement, il nécessite un grand nombre d'exemples annotés manuellement, montrant des météores dans différentes conditions et depuis différentes sources. Or, les ensembles de données annotées spécifiques aux météores sont rares et difficiles à constituer car ce sont de petits objets, ce qui rend la tâche d'entraînement de modèles de détection particulièrement ardue.

L'objectif de notre projet est de détecter des météores sur des images All Sky de manière robuste. Nous commencerons par présenter d'une manière générale, la détection de météore, son état de l'art. Nous passerons ensuite a la partie conception, ou nous présenterons notre architecture de détection. Nous terminerons par la partie réalisation, ou nous présenterons los expérimentations sur les architectures proposées.

Chapitre 1

Introduction

Rares sont les personnes qui n'ont jamais vu de phénomènes lumineux traversant le ciel à une vitesse remarquable et d'une beauté inégalée. Ces phénomènes célestes peuvent provenir d'une multitude de sources, y compris de l'espace proche (astéroïdes, comètes et débris de satellites) et de l'espace extra-atmosphérique (au-delà de notre système solaire).

FIGURE 1.1 – Traînées de météores lumineux

Ces objets peuvent être détectés lorsqu'ils pénètrent dans l'atmosphère terrestre par plusieurs méthodes, notamment les signaux sismiques, l'acoustique, les signaux à très basse fréquence (VLF) et les radars. Cependant, la méthode la plus efficace pour détecter ces objets est l'observation directe du phénomène à l'aide d'une caméra pointée vers le ciel.

FIGURE 1.2 – Caméra Allsky de type Fripon

1.1 Motivation

Le suivi des météores peut fournir des informations cruciales sur les phénomènes qui se produisent dans l'espace. La localisation d'un événement et de son point de rentrée, appelé radiant, ainsi que sa vitesse apparente et son inclinaison, permettent de déduire l'origine de l'objet, grâce à une étude dynamique complémentaire. Si cet objet poursuit sa trajectoire dans l'atmosphère et que des fragments tombent au sol, une analyse pourrait révéler des informations sur l'origine et les conditions d'évolution de l'objet, ce qui permettrait de mieux comprendre l'évolution de l'univers.

FIGURE 1.3 – Erg Chech, une météorite tombée en Algérie

Les données vidéo fournies par les caméras Allsky constituent une méthode plus fiable de détection des météores. Cette méthode est particulièrement utile pour détecter les météores pendant les heures nocturnes, car elle permet de capturer les traînées lumineuses produites par les météores lorsqu'elles se consument dans l'atmosphère. Les données vidéo des caméras Allsky peuvent éventuellement fournir d'autres informations sur les caractéristiques du météore (par exemple la vitesse). Par la suite, la combinaison de plusieurs sources de données de détection peut nous aider à obtenir des informations plus précises sur ces événements.

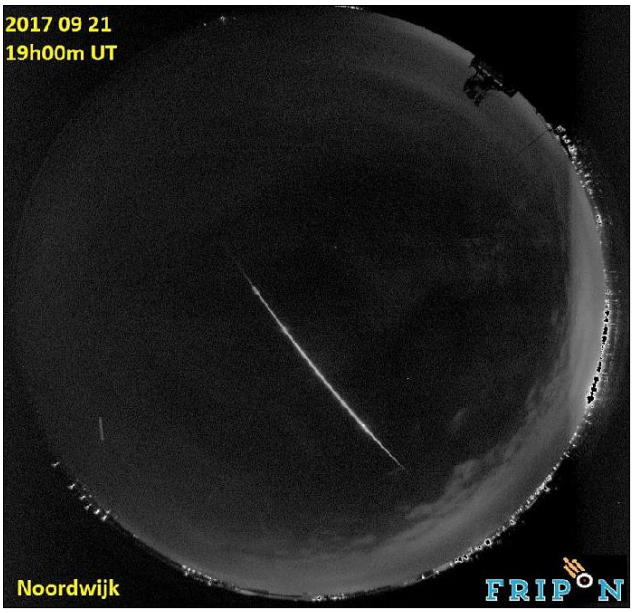

FIGURE 1.4 – Météore capturé par une caméra Allsky du Réseau Fripon [28]

1.2 Presentation du CRAAG :

1.2.1 Historique

Le CRAAG est issu de la fusion de deux institutions établies à l'époque coloniale : l'Observatoire d'Alger, fondé en 1890, et l'Institut de Physique du Globe d'Alger (IMPGA), créé en 1931. L'Observatoire d'Alger était notable pour son activité scientifique intense, se distinguant par sa participation au programme international de la Carte du Ciel de 1909 à 1925, la découverte de 64 astéroïdes, dont l'un nommé (859) Bouzaréah, ainsi que ses contributions à la détermination du temps et à l'enrichissement du catalogue fondamental. L'IMPGA, quant à lui, avait pour missions de promouvoir l'étude géophysique en Algérie et d'assurer la surveillance sismique du territoire. En 1980, le Centre National en Astronomie, Astrophysique et Géophysique (CNAAG) a été créé, regroupant les activités de l'Institut de Météorologie et de Physique du Globe d'Algérie et de l'Observatoire d'Alger, sous la tutelle de l'Office National de la Recherche Scientifique (ONRS). Ce regroupement visait à rétablir l'importance de ces deux domaines scientifiques et à garantir le développement futur des études géophysiques et astronomiques en Algérie. En 1985, la recherche scientifique a reçu un nouvel élan grâce à l'attention des pouvoirs publics, qui ont décidé de créer des Centres de Recherche par le décret 83-521. C'est ainsi que le CRAAG a été créé, héritant des responsabilités de l'ancien CNAAG tout en bénéficiant d'une organisation et d'une gestion spécifiques aux Centres de Recherche.

1.2.2 Que-est-ce que'est le CRAAG ?

Le Centre de Recherche en Astronomie Astrophysique et Géophysique CRAAG, est un Établissement Public à Caractère Scientifique et Technologique (EPST), régi par le décret 20-06 du Février 2006. Il fait partie du Haut Commissariat de la Recherche en 1990 et fût placé en 1991 sous l'autorité du Ministère de l'Intérieur. Son siège social est à la Route de l'Observatoire boîte Postale N°63, Bouzaréah, Alger, Algérie. Le CRAAG a plusieurs missions :

- Mission de recherche dans les domaines de l'Astrophysique et de la Géophysique.
- Mission de service public dans le domaine de la surveillance sismique du territoire.
- Mission auprès des secteurs Socio-économiques.

FIGURE 1.5 – Logo du CRAAG

1.2.3 Organisme Interne du CRAAG :

Le CRAAG est organisé en 3 services administratifs, 2 départements et 6 divisions, qui sont les suivants :

- LES SERVICES ADMINISTRATIFS :
  - Service du personnel et de la formation.
  - Service du budget et de la comptabilité.
  - Service des moyens généraux.
- LES DÉPARTEMENTS
  - Département des relations extérieures et de la valorisation des résultats de la recherche.
  - Département des observations géophysiques et astrophysiques.
- LES DIVISIONS DE RECHERCHE
  - La Division astrophysique solaire.
  - La Division physique stellaire et hautes énergies.
  - La Division de physique du globe.
  - La division de géophysique de subsurface.
  - La Division des études sismologiques.
  - La Division aléas et risques géologiques.

1.3 Problématique et besoin :

La détection et le suivi des météores sont d'une importance capitale pour l'acquisition d'informations sur l'évolution de l'univers. Il est donc primordial de déterminer les moyens les plus efficaces pour les repérer. La surveillance vidéo du ciel par des caméras Allsky reste

la méthode optimale à cette fin. Cependant, le développement d'un logiciel de suivi reste une entreprise difficile, en raison des caractéristiques opérationnelles de ces objets et du volume de données vidéo à gérer. L'objectif de ce travail est de proposer une approche robuste d'apprentissage profond pour la détection des météores, en prenant en compte tous les aléas de la surveillance, à savoir les phénomènes non météoriques comme les oiseaux et les avions et d'autres petits objets mobiles qui sont difficiles à identifier à l'aide de caméras Allsky et qui peuvent interférer dans la détection, et aussi le bruit de lumière.

1.4 Conclusion

En conclusion, la détection des météores est un aspect crucial de l'astronomie qui fournit des informations précieuses sur la composition et la distribution des météores dans notre système solaire. La détection des météores à l'aide de données vidéo provenant de caméras Allsky est un élément essentiel de cette recherche, offrant une approche multimodale de la compréhension des phénomènes célestes.

Chapitre 2

État de l'art

2.1 Introduction

Ce chapitre examine l'état de l'art des méthodes de détection de météores sur des images. Nous débuterons par une revue des travaux sur la détection de météores, puis nous explorerons les avancées en détection d'objets via le deep learning, en mettant l'accent sur les réseaux de neurones convolutifs (CNN) et les transformers. Enfin, nous discuterons des défis spécifiques à la détection de petits objets et des solutions pour les surmonter.

2.2 Détection de météores

La détection et la classification des météores est une tâche difficile qui a suscité beaucoup d'attention ces dernières années en raison du potentiel des météorites à fournir des informations précieuses sur les débuts du système solaire et la formation des planètes. La détection des météores à l'aide de caméras AllSky est une approche prometteuse, qui exploite les capacités des modèles d'apprentissage profond pour identifier et classer ces événements célestes.

2.2.1 Définitions

- La détection des météores consiste à identifier et à suivre les météoroïdes lorsqu'ils pénètrent dans l'atmosphère terrestre et deviennent visibles sous forme de météores.

- Dans le contexte de la détection d'objets, la détection de météores vise à automatiser le processus d'identification et de suivi des météores dans les données vidéo ou les images.

1. **Boule de feu, météore, météorite, météoroïde :** Il est essentiel de faire la différence entre les boules de feu, les météores, les météorites et les météoroïdes dans le contexte de la détection des météores :
   — Boules de feu : Météores brillants qui apparaissent sous la forme d'un flash lumineux intense et soudain dans le ciel.
   — Météores : Le passage visible d'une lampe de poche dans l'atmosphère terrestre, produisant souvent une traînée de lumière.

— Météorites : Fragments de météoroïdes qui survivent à la traversée de l'atmosphère et atteignent la surface de la Terre.
— Météoroïdes : Petits corps rocheux ou métalliques de l'espace extra-atmosphérique susceptibles de se transformer en météores lorsqu'ils pénètrent dans l'atmosphère terrestre.

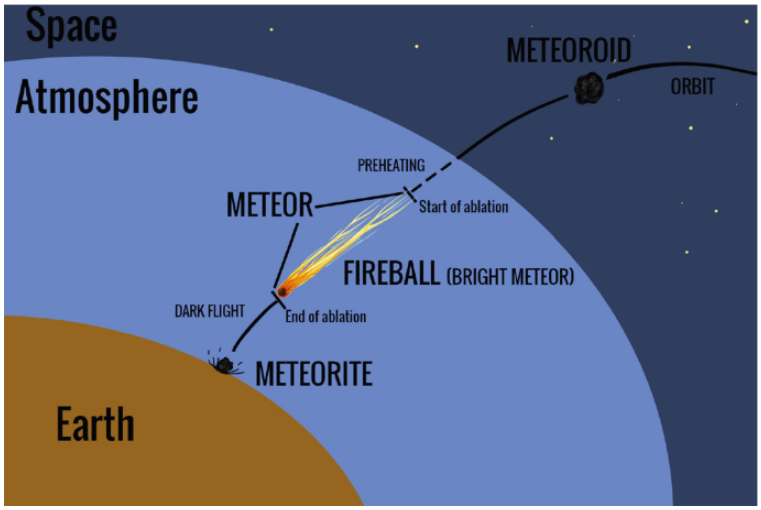

FIGURE 2.1 – Différence entre météoroïde, météore, boule de feu et météorite[3].

2. **Types de météores :** On considère deux classes de météores :
   (a) ceux qui sont censés se produire à une période spécifique de l'année parce qu'ils sont associés à des flux de météoroïdes.
   (b) Ceux qui sont sporadiques et n'ont pas de schéma périodique discernable[47].

2.2.2 Caractéristiques

Les météores présentent diverses caractéristiques qui facilitent leur identification et leur classification :

— **Luminosité** : Les météores peuvent varier en luminosité, certains produisant des éclairs intenses, connus sous le nom de boules de feu. connus sous le nom de boules de feu.
— **Vitesse** : Les météores se déplacent à grande vitesse dans l'atmosphère, créant des traînées de lumière, allant de 11 à 78 km/s s'ils proviennent de notre système solaire, et encore plus vite s'ils viennent de l'au-delà.
— **Trajectoire** : La trajectoire d'un météore peut fournir des informations précieuses sur son origine et sa composition.
— **Durée** : les météores peuvent durer d'une fraction de seconde à plusieurs secondes, en fonction de leur taille et de leur vitesse.
— **Direction** : les météores peuvent provenir de différentes directions, y compris de l'horizon et du zénith.

2.2.3 Types de météorites et leur importance scientifique

Les météorites sont des fragments d'astéroïdes ou d'autres corps célestes tombés sur Terre. Elles peuvent être classées en trois catégories principales en fonction de leur composition chimique et de leur minéralogie : les météorites pierreuses, les météorites de fer et les météorites pierreuses-ferreuses[54].

1. **Météorites de fer** : Ces météorites sont principalement composées de minéraux de fer et de nickel. Elles sont relativement rares, représentant environ 5% de l'ensemble des météorites. [54].
2. **Météorites pierreuses** : Ces météorites sont principalement composées de minéraux silicatés. Il s'agit du type de météorite le plus courant, représentant environ 94% de l'ensemble des météorites. [54].
3. **Météorites pierreuses-ferreuses** : Ces météorites contiennent à la fois des composants métalliques et silicatés. Elles sont relativement rares, représentant environ 1% de l'ensemble des météorites. [54].

La composition chimique des météorites fournit des informations précieuses sur la formation et l'évolution du système solaire. Elles contiennent des oligo-éléments et des isotopes qui permettent aux scientifiques de dater leur formation et d'étudier les processus qui se sont produits au début du système solaire [54].

FIGURE 2.2 – Types de Météorites[9].

2.2.4 Importance en spectroscopie et en photométrie

L'étude des météorites par spectroscopie et photométrie est cruciale pour comprendre leur composition et leur origine. La spectroscopie consiste à analyser la lumière émise ou absorbée par une substance, tandis que la photométrie consiste à mesurer l'intensité de la lumière. [14].

1. **Spectroscopie** : La spectroscopie est utilisée pour analyser la composition chimique des météorites en identifiant les éléments présents dans leurs spectres. Pour ce faire, les spectres des météorites sont comparés à ceux de minéraux et de roches connus [14].
2. **Photométrie** : La photométrie est utilisée pour mesurer la luminosité des météorites et de leurs corps d'origine. Pour ce faire, on analyse les courbes de lumière des météorites et de leurs corps d'origine, qui fournissenft des informations sur leur taille, leur forme et leur composition [14].

L'étude des météorites par spectroscopie et photométrie a permis des avancées significatives dans notre compréhension du système solaire. Elle a également fourni des informations précieuses sur la formation et l'évolution des planètes et d'autres corps célestes [14].

2.2.5 Travaux sur la détection de météores

1. Logiciel Freeture

Freeture est un logiciel open-source de détection de météores utilisé pour surveiller le ciel à l'aide de caméras tout ciel GigE afin de détecter et d'enregistrer les étoiles filantes et les boules de feu. Il a été développé à l'origine pour le projet FRIPON (Fireball Recovery and InterPlanetary Observation Network), qui vise à couvrir toute la France avec 100 caméras fish-eye.

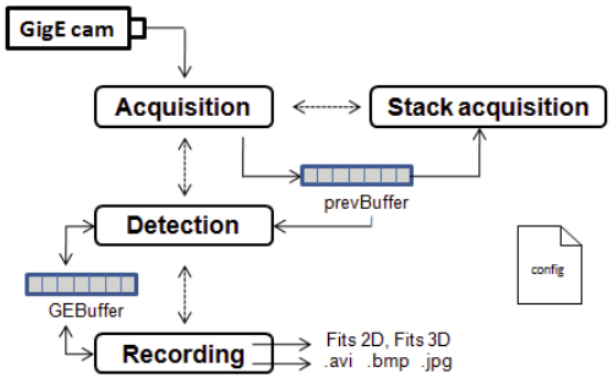

FIGURE 2.3 – Freeture Software Outline[2].

Inconvénients de la méthode :

— **Personnalisation limitée :** Le logiciel peut ne pas être facilement adaptable à des besoins spécifiques ou à des améliorations des algorithmes de détection.

— **Forte dépendance au matériel :** Les performances de Freeture peuvent dépendre fortement de la qualité et des spécifications des caméras utilisées.
— **Vitesse de traitement :** En fonction du matériel et de la quantité de données, la vitesse de traitement peut être un problème, en particulier avec les fonctions d'exposition longue et d'empilement d'images.
— **Précision de détection :** L'algorithme de détection peut être confronté à des faux positifs, en particulier dans les environnements complexes ou bruyants.
— **Configuration manuelle :** Une configuration manuelle importante peut être nécessaire pour optimiser le système en fonction des différentes conditions d'observation.

2. CAMS

CAMS [24] est un projet parrainé par la NASA et dirigé par Peter Jenniskens à l'Institut SETI. Le projet CAMS utilise l'IA pour identifier les météores dans le ciel nocturne, en les distinguant d'autres objets tels que les oiseaux, les avions et les satellites. Il analyse les images capturées par les caméras du réseau CAMS, détecte les pluies de météores et les reporte sur le portail des pluies de météores de la NASA. [32].

Inconvénients de la méthode

1. **Précision et faux positifs** : L'un des principaux défis du système CAMS est de maintenir une grande précision dans la détection des météores tout en minimisant les faux positifs. La présence de divers objets aériens peut déclencher de fausses détections, ce qui complique le processus de classification. Bien que le système utilise des algorithmes sophistiqués pour atténuer ce phénomène, la possibilité d'erreurs demeure.
2. **Traitement et stockage des données** : La manipulation et le traitement de grands volumes de données provenant de plusieurs stations de caméras nécessitent beaucoup de ressources. Bien que la transition vers la transmission et le traitement automatisés des données ait rationalisé le flux de travail, il faut encore des ressources informatiques importantes et une infrastructure solide pour gérer efficacement les données.
3. **Couverture initiale limitée** : Au départ, le réseau de caméras était limité, ce qui restreignait la zone de couverture pour la détection des météores. Bien que le projet se soit développé et qu'il ait intégré des citoyens scientifiques pour accroître sa portée, il subsiste des lacunes géographiques qui peuvent faire manquer des événements météorologiques[32].
4. **Dépendance des conditions environnementales** : La précision de la détection des météores peut être affectée par les conditions environnementales telles que les conditions météorologiques, la pollution lumineuse et le positionnement de la caméra. Ces facteurs peuvent avoir un impact sur la visibilité et la clarté des météores, ce qui rend difficile la garantie d'une qualité de détection constante sur différents sites et à différents moments [32].

3.Apprentissage Profond pour la Surveillance des Météores

Ce travail [47] présente un ensemble d'IA-météores entièrement automatisé qui détecte, analyse et distingue les différents types de météores à l'aide de réseaux de neurones convolutifs

(CNN). La méthode est capable de détecter les météores même dans des images contenant des éléments statiques tels que des nuages, la Lune et des bâtiments. L'utilisation de la cartographie d'activation de classe pondérée par gradient (Grad-CAM) permet de localiser avec précision le météore dans chaque image, ce qui améliore la précision du suivi et de la classification des météores. Les auteurs ont utilisé un ensemble de données comprenant 982 images de météores et 1050 images sans météores détectées par les stations SPMN en 2021 pour l'entraînement du modèle CNN.

Inconvénients de la méthode

1. **Susceptibilité aux faux positifs** : Le modèle peut confondre des satellites avec des météores en raison de caractéristiques visuelles similaires, comme le montre l'exemple du train de satellites Starlink de SpaceX.
2. **Problèmes liés à l'augmentation des données** : Bien que l'augmentation des données ait été utilisée pour éviter le surapprentissage, elle peut potentiellement dégrader les performances du modèle. Cela aurait pu se traduire par des performances légèrement inférieures à celles d'autres modèles, malgré une précision globale élevée.
3. **Exigence de supervision manuelle** : Même avec l'automatisation, une supervision humaine est toujours nécessaire pour filtrer les faux positifs et organiser les enregistrements, ce qui peut constituer un goulot d'étranglement.
4. **Interférence environnementale** : Les performances du modèle peuvent être affectées par des conditions environnementales telles que la présence de nuages, de la Lune, de bâtiments et d'autres obstacles susceptibles d'entraver la détection précise des météores.

4. Filtrage de la Détection de Météores en Station Unique utilisant l'Apprentissage Automatique

Ce papier [13] introduit une approche d'apprentissage automatique pour améliorer la détection des météores à partir des observations d'une seule station, qui sont sujettes à un taux élevé de faux positifs. En extrayant des caractéristiques pertinentes du mouvement des météores et en appliquant divers modèles d'apprentissage automatique, les auteurs obtiennent une précision de détection et un rappel élevés. La méthodologie réduit efficacement la charge des validateurs humains et améliore la fiabilité de la détection des météores dans les réseaux optiques de boules de feu .

Inconvénients de la méthode

1. **Taux élevé de faux positifs dans les observations d'une seule station** : La détection des météores par une seule station produit généralement un nombre élevé de faux positifs, ce qui nécessite un filtrage et une validation supplémentaires pour garantir la précision des résultats.
2. **Complexité de l'extraction des caractéristiques** : L'extraction de caractéristiques pour chaque détection et l'identification des plus pertinentes nécessitent des ressources informatiques importantes et une expertise dans le traitement des données.

3. **Nécessité de disposer de données d'entraînement diversifiées** : L'ensemble de données d'entraînement actuel n'est peut-être pas assez complet pour permettre une bonne généralisation à tous les événements météorologiques possibles. Les auteurs suggèrent d'élargir l'ensemble de données d'entraînement en utilisant des sources supplémentaires telles que la base de données FRIPON afin d'améliorer la généralisation du modèle.

5. Détection de Météores utilisant des Réseaux de Neurones Convolutionnels Profonds

Cet article [42] décrit une application d'un réseau de neurones convolutionnel profond pré-entraîné pour détecter les météores à partir d'images du ciel nocturne. L'ensemble de données est relativement petit, composé d'images étiquetées de météores et de non-météores du ciel nocturne. Des techniques telles que l'augmentation des données ont été utilisées pour créer des données artificielles, et une couche d'exclusion a été introduite pour empêcher le sur-apprentissage de l'ensemble de données d'entraînement augmentées. Les performances obtenues par les méthodes utilisant la validation croisée quintuple traditionnelle ainsi que le changement des images en noir et blanc, comparées à celles obtenues en utilisant uniquement les partitions d'entraînement et de validation et les couleurs RGB, ont atteint une précision moyenne plus élevée de 84,35%.

Inconvénients de la méthode

1. **Petit ensemble de données** : L'ensemble de données utilisé pour l'entraînement était relativement petit, composé de 310 images de météores et de 185 images de non-météores. Les petits ensembles de données peuvent limiter la capacité du modèle à se généraliser à de nouvelles données inédites, augmentant ainsi le risque de surapprentissage.
2. **Qualité des données** : L'ensemble de données contenait des données bruitées, notamment des images de mauvaise qualité, des lampadaires et d'autres figures sans rapport, ce qui pourrait avoir un impact négatif sur les performances du modèle.
3. **Limites de l'augmentation des données** : Bien que l'augmentation des données permette d'accroître artificiellement la taille de l'ensemble de données, elle ne remplace pas totalement un ensemble de données réel, vaste et diversifié, ce qui peut entraîner des biais dans le modèle formé.
4. **Dépendance sur le Transfer Learning** : Le recours à l'apprentissage par transfert à partir d'un modèle pré-entraîné (VGG16) signifie que les performances du modèle dépendent en partie des caractéristiques apprises à partir de l'ensemble de données ImageNet, qui peuvent ne pas être entièrement pertinentes pour la détection des météores.
5. **Généralisation** : Les performances du modèle, bien que raisonnablement élevées, peuvent encore avoir du mal à se généraliser à différentes conditions de ciel nocturne ou à des variations dans l'apparence des météores qui ne sont pas représentées dans l'ensemble de données d'apprentissage.

6. Détection et localisation des météores en utilisant YOLOv3 et YOLOv4

Al-Owais et al.[12] ont utilisé les algorithmes de détection d'objets YOLOv3 et YOLOv4, qui utilisent des réseaux de neurones convolutionnels, pour détecter et localiser les météores dans les images. Ils ont entraîné les modèles sur un ensemble de données équilibré et déséquilibré composé de milliers d'images de météores et d'objets non météoriques tels que des avions, des oiseaux, des insectes et des débris spatiaux. La recherche se concentre sur les capacités de détection en temps réel et sur l'amélioration de la précision de la détection.

Résultats

Le modèle YOLOv4 a obtenu le score de rappel le plus élevé 98,5%, suivi par le modèle YOLOv3 avec un score de rappel de 98%. Le modèle YOLOv4 déséquilibré a également obtenu la plus grande précision 90%. Les quatre modèles ont réussi à étiqueter les météores avec un taux de confiance supérieur à 95%.

Inconvénients de la méthode

1. **L'absence de comparaison avec d'autres algorithmes de détection d'objets** : L'étude ne compare pas les performances de YOLOv3 et YOLOv4 avec d'autres algorithmes de détection d'objets de pointe, ce qui rend difficile l'évaluation de leurs forces et faiblesses relatives.
2. **Métriques d'évaluation limitées** : L'étude se concentre sur le rappel et la précision en tant que mesures d'évaluation primaires, mais ne prend pas en compte d'autres mesures importantes telles que la précision, le score F1 ou la précision moyenne (mAP).
3. **L'absence de déploiement et de test dans le monde réel** : L'étude ne rend pas compte des performances des modèles dans des scénarios de détection de météores dans le monde réel, comme leur capacité à gérer des conditions météorologiques, des angles de caméra et des trajectoires de météores variables.
4. **Exigences de calcul élevées pour l'entraînement des modèles YOLO**.

7. Algorithmes d'apprentissage profond appliqués à la classification de la détection vidéo de météores

Ce travail [31] se concentre sur l'exploitation des algorithmes d'apprentissage profond pour détecter et classer les météores dans les données vidéo. Cette recherche améliore la précision et l'efficacité des systèmes de détection des météores, qui reposent traditionnellement sur des techniques manuelles ou semi-automatiques.

Inconvénients et limitations de la méthode

1. **Variabilité limitée des données** : L'ensemble des données est principalement constitué de vidéos provenant d'une seule source, ce qui peut ne pas refléter toute la diversité des apparitions de météores et des conditions environnementales. Cette limitation pourrait affecter la capacité de généralisation des modèles à différents contextes d'observation des météores.
2. **Généralisation à d'autres phénomènes** : Bien que les modèles aient montré de bonnes performances sur les données d'essai, leur capacité à distinguer les météores

d'autres phénomènes similaires tels que les avions, les satellites ou les éclairs n'a pas été évaluée de manière approfondie. Cela pourrait conduire à des faux positifs dans des applications réelles où ces phénomènes sont courants.

8. Application des réseaux de neurones convolutionnels à l'automatisation d'un pipeline de détection de météores

Ce travail [19] présente un pipeline entièrement automatisé pour la détection des météores provenant de l'Observatoire canadien automatisé des météores (CAMO). Le nouveau pipeline vise à éliminer le goulot d'étranglement causé par la nécessité d'une intervention humaine dans la vérification des événements météorologiques, améliorant ainsi l'efficacité de la détection des météores et du traitement des données. L'ensemble de données pour l'entraînement et le test du CNN comprenait 50,745 échantillons étiquetés manuellement et collectés à partir des données de la caméra d'influx de CAMO. Le CNN a atteint un taux de précision impressionnant de 99,8%.

Limitations et Inconvénients de la méthode

Malgré sa grande précision, le pipeline présente certaines limites :

1. **Faux négatifs :** Certains météores de faible intensité ont été incorrectement étiquetés comme n'étant pas des météores, ce qui suggère que le CNN peut manquer des événements très subtils.
2. **Ajustements de la taille du noyau :** La nécessité d'ajuster manuellement la taille des noyaux et les fonctions d'activation indique que le pipeline peut nécessiter une optimisation et un réglage supplémentaires pour différents ensembles de données ou conditions d'observation.
3. **Généralisation à d'autres ensembles de données :** Alors que le pipeline fonctionne bien sur les données CAMO, sa performance sur les données provenant d'autres observatoires ou de conditions environnementales différentes n'est pas entièrement évaluée. D'autres tests et validations sont nécessaires pour garantir sa robustesse et sa généralisation.

9. Classification d'objets sur les données vidéo de météores et de phénomènes similaires aux météores

Sennlaub et Hofmann [52], présentent une étude complète sur la classification des météores et des phénomènes similaires à l'aide de techniques avancées d'apprentissage automatique. Leur travail porte sur le manque de données d'entraînement de haute qualité disponibles pour la reconnaissance précise des météores traversant l'atmosphère terrestre. Ils ont développé l'ensemble de données NightSkyUCP, qui se compose de 20 000 événements également répartis entre les événements météoriques et non météoriques. L'ensemble de données NightSkyUCP comprend des données vidéo, des piles d'images recadrées et des données de mouvement, ce qui constitue une ressource riche pour l'entraînement de modèles d'apprentissage automatique.

Limitations et Inconvénients de la méthode

1. **Biais de sélection :** L'ensemble de données se compose principalement de météores et de sous-classes de non-météores.
2. **Classes déséquilibrées :** La classe des non-météores est moins représentée, rendant la classification de ses sous-classes difficile.
3. **Plage dynamique :** Les vidéos varient considérablement en taille et en durée, ce qui complique le processus d'entraînement du modèle.
4. **Précision de l'extrapolation :** La précision diminue pour les objets plus rapides ou ceux avec des vitesses changeantes, rendant la prédiction des trajectoires de météores moins fiable dans ces cas.
5. **Biais dans les données :** L'ensemble de données peut être biaisé car il n'inclut pas tous les événements ressemblant à des météores, en particulier ceux exclus par l'algorithme de détection.

10. Contribution à l'étude et à la réalisation d'un système pour la détection automatique et le suivi des météores lumineux

Ce travail [16] présente une approche globale de la détection des météores, comprenant des étapes de prétraitement telles que la subdivision, la compression et la différenciation des images, suivies de phases de détection et de post-traitement pour la validation et l'association des événements, tout en relevant des défis tels que la sensibilité des paramètres et les complexités de l'arrière-plan dynamique.

Inconvénients et limitations de la méthode

— **Sensibilité aux paramètres :** L'efficacité de la méthode dépend fortement du réglage des paramètres qui, s'ils sont imprécis, peuvent entraîner des faux positifs ou des détections manquées.

— **Défis liés aux arrière-plans dynamiques :** Les hypothèses relatives aux arrière-plans statiques peuvent limiter les performances dans les environnements dynamiques ou avec des nuages en mouvement, ce qui peut entraîner des fausses détections ou des événements manqués.

— **Complexité de calcul :** Les étapes collectives de prétraitement, de détection et de post-traitement augmentent le temps de traitement, en particulier pour les applications en temps réel ou les images à haute résolution.

— **Problèmes d'adaptabilité :** Bien que le seuillage adaptatif améliore la robustesse, il peut s'avérer difficile dans des conditions extrêmes, ce qui nécessite des ajustements manuels ou un prétraitement supplémentaire.

Les tableaux 2.1,2.2 et 2.3 présentent une comparaison détaillée des travaux de l'état de l'art sur la détection de météores.

TABLE 2.1 – Comparaison des travaux de l'état de l'art

État de l'art	Pipeline IA CAMS	Apprentissage Profond pour la Surveillance des Météores	Détection de Météores utilisant des Réseaux de Neurones Convolutionnels Profonds	Filtrage de la Détection de Météores en Station Unique utilisant l'Apprentissage Automatique
Type de Technique	Pipeline IA avec Apprentissage Automatique et Apprentissage Profond (CNN & LSTM)	Réseau de Neurones Convolutionnel (CNN)	Réseau de Neurones Convolutionnel (CNN)	Apprentissage Automatique (ML)
Type de Détection	Détection et Classification	Détection et Classification	Détection et Classification	Classification (Météore vs. Non-Météore)
Ensemble de Données Utilisé	Données du ciel de divers sites avec transmission FTP	Images de SPMN (Réseau Espagnol de Météores)	Jeu de données personnalisé EXOSS	Données MOROI (2017-2020)
Échantillons	Non spécifié, jeu de données multi-années de divers sites	982 images de météores, 1,050 images de non-météores	310 images de météores, 185 images de non-météores	8,086 événements (1,164 véritables météores et 6,922 non-météores)
Étapes de Traitement	Six étapes : traitement local, récupération des données, scripts Python, calcul des coïncidences, clustering, visualisation	Augmentation des données, apprentissage par transfert avec VGG16, Grad-CAM pour la visualisation	Augmentation des données, apprentissage par transfert avec VGG16	Extraction de caractéristiques (24 caractéristiques), entraînement du modèle d'apprentissage automatique et classification
Résultats	Haute précision et rappel dans la classification, automatisation extensive, visualisation sur interface web	Haute précision (98%), robustesse du modèle testée dans diverses conditions	Précision de 84,35% utilisant la validation croisée à cinq reprises, améliorée avec l'augmentation des données	Haute précision (98,2%) et rappel (96%), validation humaine réduite
Inconvénients	Nécessite des ressources computationnelles significatives, potentiel de faux positifs, interférences environnementales	Forte demande computationnelle, quelques faux positifs, besoin de données plus diversifiées	Petit jeu de données, données bruyantes, dépendance à l'apprentissage par transfert, intensif en calculs	Coût computationnel élevé pour l'extraction de caractéristiques, diversité limitée des données d'entraînement

TABLE 2.2 – Comparaison des travaux de l'état de l'art

État de l'art	**Classification d'objets sur les données vidéo de météores et de phénomènes similaires aux météores**	**Application des réseaux de neurones convolutionnels à l'automatisation d'un pipeline de détection de météores**	**Algorithmes d'apprentissage profond appliqués à la classification de la détection vidéo de météores**	**Détection et localisation des météores en utilisant YOLOv3 et YOLOv4**
Type de Technique	SVM, Extraction de caractéristiques	CNN, Apprentissage par transfert	CNN	YOLOv3, YOLOv4
Type de Détection	Classification d'objets	Détection de météores	Détection de météores	Détection et localisation des météores
Jeu de Données Utilisé	Données vidéo de plusieurs stations d'observation de météores	Enregistrements vidéo de l'American Meteor Society	Images vidéo de caméras grand angle	Enregistrements vidéo de plusieurs caméras grand angle
Échantillons	Étiquetage extensif pour assurer une représentation diversifiée	Milliers d'images	Non spécifié, mais inclut des apparitions diverses de météores	Données annotées avec étiquettes météore et non-météore
Étapes de Traitement	Extraction de caractéristiques, Classification SVM	Extraction d'images, Augmentation des données, Entraînement CNN	Prétraitement des données, Augmentation des données, Entraînement CNN	Annotation des données, Entraînement YOLO, Évaluation
Résultats	Classification efficace des météores et des phénomènes similaires aux météores	Détection automatisée des météores avec une précision améliorée	Détection efficace des météores avec une bonne précision et un bon rappel	Détection et localisation en temps réel avec haute précision et rappel
Inconvénients	Évolutivité limitée avec l'augmentation du volume de données, caractéristiques subtiles potentiellement manquées	Surapprentissage potentiel, bruit dans l'annotation manuelle	Difficultés avec les faux positifs, forte dépendance à la qualité du jeu de données	Exigences computationnelles élevées, défis dans la détection des petits/-faibles météores

État de l'art	**Contribution à l'étude et à la réalisation d'un système pour la détection automatique et le suivi des météores lumineux**	**Freeture**
Type de Technique	Approche globale (Subdivision de l'image, Compression, Différentiation, Détection, Post-traitement)	Logiciel open-source de détection de météores, utilisant du traitement d'image
Type de Détection	Détection et suivi des météores	Détection de météores
Jeu de Données Utilisé	Images de caméras grand angle	Images de caméras grand angle
Échantillons	Échantillons d'images traités à travers diverses étapes	Images capturées par des caméras grand angle GigE
Étapes de Traitement	Subdivision de l'image, Compression, Différentiation, Seuil, Formation d'événements locaux, Liaison, Validation, Association d'événements globaux, Post-traitement	Acquisition d'images, Sortie au format FITS, Acquisition en longue exposition, Empilement d'images, Algorithme de détection
Résultats	Détection et suivi efficaces des météores lumineux avec validation et association d'événements complets	Détection et enregistrement efficaces des météores et des bolides, avec des capacités pour la longue exposition et l'empilement d'images
Inconvénients	Sensibilité aux paramètres, défis liés aux arrière-plans dynamiques, complexité computationnelle, préoccupations d'adaptabilité	Personnalisation limitée, forte dépendance au matériel, problèmes de vitesse de traitement, préoccupations de précision de détection, configuration manuelle

TABLE 2.3 – Comparaison des travaux de l'état de l'art

2.2.6 Limitations et Difficultés liées la Détection de Météores

Malgré les progrès réalisés dans les techniques de détection des météores, plusieurs défis persistent dans ce domaine. Il s'agit notamment des problèmes liés au bruit de fond, aux différents niveaux de luminosité des météores, à la distinction entre les météores et d'autres phénomènes célestes, et à la nécessité de disposer de capacités de traitement en temps réel pour capturer efficacement les événements transitoires.

1. **Petite taille :** Les météores sont souvent petits et apparaissent comme de faibles points lumineux, ce qui les rend difficiles à détecter sur l'arrière-plan.
2. **Faux positifs et bruit :** Les météores doivent être distingués des faux positifs causés par les satellites, les avions, les hélicoptères, les drones, les oiseaux, les éclairs ou les sources de lumière artificielle. Les caméras tout ciel peuvent être sensibles au bruit et aux artefacts, ce qui entraîne des détections faussement positives qu'il convient d'atténuer. Cela peut s'avérer particulièrement difficile dans les environnements à faible contraste, où les météores sont difficiles à distinguer du bruit de fond.
3. **Temps d'exécution :** Le principal inconvénient des algorithmes actuels de suivi des météores est la durée d'exécution nécessaire à l'analyse des images vidéo haute définition 1080p. [47]
4. **Flou de mouvement et faible contraste :** Les météores se déplacent rapidement dans le ciel, ce qui entraîne un flou de mouvement sur les images ou les vidéos, qui peut nuire à la précision de la détection. En outre, le faible contraste entre les météores et le ciel nocturne ou l'arrière-plan peut les rendre difficiles à distinguer.
5. **Conditions d'éclairage variables :** Les changements d'éclairage, tels que les nuages, le clair de lune ou la pollution lumineuse, peuvent affecter la visibilité des météores et compliquer leur détection.
6. **Nature dynamique des météores :** Le mouvement rapide et les trajectoires imprévisibles des météores posent un problème de détection et de suivi précis.
7. **Complexité de calcul :** La complexité de calcul du traitement de grandes quantités de données provenant de caméras couvrant l'ensemble du ciel peut être importante, nécessitant un matériel puissant et des algorithmes sophistiqués pour traiter les données de manière efficace.
8. **Qualité des données et prétraitement :** La qualité des données collectées par les caméras all-sky peut être affectée par divers facteurs tels que la résolution de la caméra, la fréquence d'images et les conditions environnementales. Des techniques de prétraitement sont nécessaires pour s'assurer que les données sont adaptées à l'analyse.
9. **Interprétation des résultats :** L'interprétation des résultats des algorithmes de détection des météores peut s'avérer difficile en raison de la complexité des données et de la nécessité de faire la distinction entre les météores et les autres phénomènes célestes.
10. **Traitement en temps réel :** La nécessité de disposer de capacités de traitement en temps réel pour capturer efficacement les événements transitoires peut constituer une limitation importante dans le développement des systèmes de détection des météores.

2.3 Ensembles de données de détection des météores

L'entraînement et l'évaluation des modèles d'apprentissage automatique, en particulier dans le domaine de la détection des météores, s'appuient fortement sur des ensembles de données. Ces ensembles de données fournissent un échantillon représentatif des données du monde réel qui permet aux modèles d'apprendre les caractéristiques des météores et de les

distinguer du bruit de fond. Des ensembles de données diversifiés et de haute qualité sont essentiels pour faire avancer la recherche dans le domaine de la détection des météores. Ils permettent aux modèles de traiter un large éventail d'apparitions de météores et de conditions environnementales, ce qui se traduit en fin de compte par des performances plus robustes et généralisables.

2.3.1 Exigences relatives aux ensembles de données de détection des météores :

- **Variété dans l'apparence des météores :** Les ensembles de données doivent comprendre des météores de tailles, de formes et de niveaux de luminosité différents. Cela permet de s'assurer que les modèles peuvent détecter les météores de faible intensité, les courtes traînées et les grands bolides (boules de feu).
- **Conditions environnementales diverses :** Les observations de météores dans le monde réel sont influencées par des facteurs tels que la pollution lumineuse, la couverture nuageuse et les interférences atmosphériques. Les ensembles de données doivent capturer cette variabilité pour former des modèles qui fonctionnent efficacement dans diverses conditions d'observation.
- **Haute résolution :** Les images ou vidéos à haute résolution sont essentielles pour capturer les détails fins des météores.
- **Vérité terrain :** Les annotations sont essentielles pour les approches d'apprentissage supervisé. Les annotations doivent identifier avec précision la présence et l'emplacement des météores dans chaque image, y compris les boîtes englobantes précises ou les masques de segmentation. La qualité et la cohérence des annotations ont un impact direct sur les performances du modèle.

2.3.2 Ensembles de données existants sur la détection des météores

Plusieurs ensembles de données accessibles au public ont été utilisés dans la recherche sur la détection des météores :

- **CAMS (Cameras for Allsky Meteor Surveillance) :** CAMS est un projet qui utilise l'apprentissage automatique pour détecter les comètes à longue période grâce à l'analyse des données météorologiques [32]. Cet ensemble de données fournit des images à haute résolution capturées par des réseaux de caméras couvrant l'ensemble du ciel à différents endroits. Il offre une bonne diversité dans l'apparence des météores, mais peut présenter des limites en termes de variabilité environnementale (il est nécessaire de contacter les auteurs pour y avoir accès).

FIGURE 2.4 – CAMS Turkey Station in YOZGAT [32].

— **FRIPON (Fireball Recovery and InterPlanetary Observation Network) :** FRIPON est un réseau mondial qui suit les météoroïdes entrants à l'aide d'une combinaison de caméras et de récepteurs radio. L'ensemble des données comprend plus de 4000 météoroïdes détectés entre 2016 et 2020, avec des informations détaillées sur leurs orbites et leurs propriétés physiques [23]. Bien qu'elle offre une couverture étendue, la résolution n'est peut-être pas idéale pour capturer les météores de faible intensité[28].

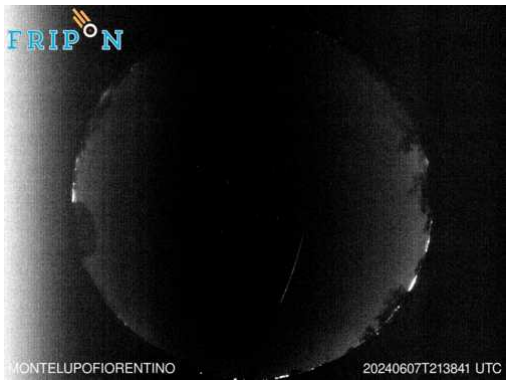

FIGURE 2.5 – Example of the Stacked Detection Captured by FRIPON Station in Montelupo Fiorentino[28].

— **MAIA (Meteor Automatic Imager and Analyzer) :** Cet ensemble de données offre un bon équilibre entre la taille et la diversité. Il comprend des images présentant diverses apparences de météores capturées par des systèmes automatisés de caméras tout ciel. Toutefois, la qualité des annotations peut varier en fonction de la version spécifique de l'ensemble de données [37].

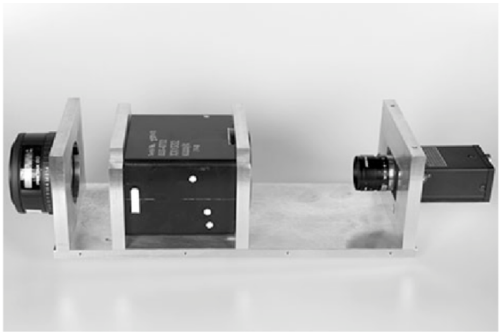

FIGURE 2.6 – System MAIA inner housing with installed components. From left-objective lens, image intensifier, camera lens and digital camera [37].

— **EXOSS (Exploring the Southern Sky) :** EXOSS est un projet scientifique citoyen dédié à la surveillance des météores. EXOSS exploite plus de 50 stations équipées de caméras CCTV peu coûteuses qui observent en permanence le ciel nocturne, capturant automatiquement des images potentielles de météores, de météorites et d'objets non météoriques. L'ensemble de données, qui comprend 400 images, présente des difficultés en raison de sa taille relativement petite et de la présence de données bruitées, telles que des images de mauvaise qualité, des lampadaires et des figures sans rapport avec le sujet[42].

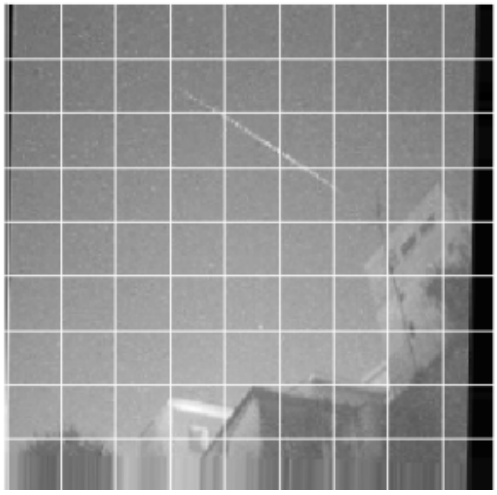

FIGURE 2.7 – Black and White figure of a Meteor Captured by the EXOSS Station [42].

— **MOROI (Meteorites Orbits Reconstruction by Optical Imaging Network) :** MOROI est un réseau de caméras utilisé pour détecter et suivre les météoroïdes. Le jeu de données comprend des orbites de météoroïdes reconstruites à partir de données d'imagerie optique entre 2017 et 2020 [13].

Il est important de noter que chaque ensemble de données a ses propres forces et limites : CAMS et EXOSS offrent un bon équilibre en termes de taille et de diversité, mais risquent de ne pas couvrir tout le spectre des conditions environnementales. Fripon offre une couverture étendue mais pourrait être limité en termes de résolution pour capturer les météores de faible intensité. MOROI offre des informations spectrales mais peut être limité en termes de volume de données et de conditions d'observation. Le Tableau 2.4 compare les différents ensembles de données de météores.

TABLE 2.4 – Comparaison des ensembles de données existants pour la Détection des Météores

Dataset	**Taille**	**Résolution**	**Annotations**	**Forces**	**Limites**
CAMS	Grande (nécessite de contacter les auteurs)	Haute	Peut varier	Images haute résolution, Apparences diversifiées des météores	Variabilité environnementale limitée, Restrictions d'accès
FRIPON	Moyenne (4000+ météores)	Modérée	Étendues (orbites, propriétés physiques)	Large couverture	Résolution inférieure pouvant manquer les météores faibles
MAIA	Moyenne	Modérée	Variable (dépend de la version)	Bon équilibre entre taille et diversité	Qualité d'annotation incohérente
EXOSS	Petite	Basse	Limitées	Science citoyenne - large couverture du ciel	Taille très petite, Données bruyantes (qualité médiocre, pollution lumineuse)
MOROI	Moyenne	Modérée	Données de trajectoire (orbites)	Informations orbitales détaillées	Période limitée (2017-2020)

2.3.3 Défis

Malgré les progrès réalisés dans les ensembles de données de détection des météores, plusieurs défis et limitations subsistent :

- **Taille et diversité limitées :** De nombreux ensembles de données sont limités en taille et en diversité, ce qui peut entraîner un sur-apprentissage et une réduction de la généralisation des modèles.
- **Annotations de mauvaise qualité :** Une mauvaise qualité d'annotation peut entraîner une réduction des performances du modèle.
- **Défis spécifiques de la détection des météores :** La détection des météores pose des défis uniques, tels que la nécessité de gérer des conditions environnementales variables et l'exigence d'annotations de haute qualité.

2.4 Détection d'objets :

La détection d'objets (DO) est une technique de vision par ordinateur, qui consiste à trouver toutes les instances d'objets d'une ou plusieurs classes d'objets (personne, voiture, chien, chat...) sur une image numérique et de les classifier et ce, indépendamment de l'échelle, de l'emplacement, de la pose, de la vue par rapport à la caméra, des occlusions partielles et des conditions d'éclairage. Ainsi, les deux tâches les plus importantes de la DO sont :

— Localisation d'objets.
— Classification de ces objets et estimations de leur taille avec une boîte englobante (Bounding Box).

2.4.1 Méthodes de DO :

Suite aux avancées du Deep Learning dans la détection d'objets, on ne va que s'interesser aux méthodes basées sur l'apprentissage profond.

Détecteurs basés sur l'apprentissage profond :

Durant les années 2010-2012, le progrès de la DO aurait quelque peu stagné et serait même devenu saturé. Mais ce n'était qu'une question de temps avant que l'informatique ait connu la renaissance des réseaux de neurones convolutionnels (Convolutional Neural Networks CNN) [38]. R. Girshick et al. ont pris l'initiative de sortir de l'impasse en 2014 en proposant le modèle Regions with CNN features (RCNN) [30]. Depuis, la détection d'objets a commencé à évoluer à une vitesse sans précédent. Les algorithmes de DO à base de CNN sont divisés en 2 types : ceux qui effectuent la détection en deux phases, et d'autres en une seule phase :

a. Détecteurs en 2 phases :

Ces méthodes perçoivent les deux tâches de détection comme deux phases séparées :

1. Region Proposals (proposition de région) est une méthode utilisée pour déterminer la localisation des objets.
2. Classification des objets suite au données extraites durant la première phase et régression des boîtes englobantes.

— **RCNN :** Le concept derrière RCNN est direct : il commence par générer un ensemble de propositions d'objets (boîtes de candidats) par la méthode de recherche sélective [56]. Chaque proposition est ensuite redimensionnée à une taille fixe et introduite dans un modèle CNN pré-entraîné sur ImageNet, tel qu'AlexNet [38], pour extraire ses caractéristiques. Enfin, des classificateurs SVM linéaires sont utilisés pour prédire la présence d'un objet dans chaque région et pour identifier leurs catégories. RCNN a significativement amélioré les performances sur VOC07 (Ensemble de données Pascal VOC 2007 [27]). Cependant, ses inconvénients incluent des calculs redondants sur un

grand nombre de propositions superposées (plus de 2000 boîtes par image), ce qui ralentit considérablement la vitesse de détection (14s par image avec GPU).

— **Fast RCNN :** R. Girshick et al ont introduit Fast RCNN [29], une amélioration notable de RCNN [30] en 2015. Fast RCNN permet l'entraînement simultané d'un détecteur et d'un régresseur de boîtes englobantes dans un cadre de réseau unifié. Sur VOC07, Fast RCNN a augmenté le mAP de 58,5% (RCNN) à 70,0%, tout en améliorant la vitesse de détection de plus de 200 fois par rapport à RCNN. Bien que Fast RCNN ait gardé les avantages de RCNN et augmentant la vitesse, cette dernière reste toute aussi limitée par la détection des propositions.

— **Faster RCNN :** Peu après Fast RCNN, S. Ren et al. ont présenté Faster RCNN [51] en 2015. Faster RCNN est le premier détecteur d'objets en temps quasi réel grâce à son mAP COCO@.5=42,7% (Ensemble de données MSCOCO [40]) et mAP VOC07=73,2%, avec une vitesse de 17 FPS utilisant ZF-Net [62]. La contribution majeure de Faster RCNN réside dans l'introduction d'un Réseau de Propositions de Régions (RPN), permettant de générer des propositions de régions presque sans coût de calcul. Cette évolution a permis d'intégrer progressivement tous les blocs individuels d'un système de détection d'objets dans un cadre d'apprentissage unifié, de R-CNN à Faster RCNN.

— **Feature Pyramid Networks (FPN) :** En 2017, T.-Y. Lin et al. ont introduit les Feature Pyramid Networks (FPN) [39]. Avant FPN, la plupart des détecteurs basés sur l'apprentissage profond se concentraient uniquement sur les cartes de caractéristiques des couches supérieures du réseau. Bien que ces caractéristiques soient utiles pour la reconnaissance des catégories, elles ne permettent pas une localisation précise des objets à toutes les échelles. FPN a introduit une architecture ascendante avec des connexions latérales pour construire une représentation sémantique à plusieurs échelles. Cette approche exploite naturellement la pyramide de caractéristiques formée par un CNN, améliorant ainsi significativement la précision des détecteurs d'objets comme Faster RCNN (mAP COCO@.5=59,1%). FPN est maintenant largement adopté comme composante essentielle dans de nombreux détecteurs modernes.

b. Détecteurs en une phase :

Les détecteurs en deux phases peuvent facilement atteindre une haute précision sans aucun artifice, mais sont lourds et complexes. Les détecteurs en une phase, compte à eux, peuvent retrouver tous les objets en une seule étape d'inférence. Ils sont utilisés sur les petits appareils pour leurs fonctions en temps réel, mais leurs performances diminuent sensiblement lorsqu'il s'agit de détecter des objets denses et petits.

You Only Look Once (YOLO) : Le premier algorithme de détection en une seule phase est YOLO [49]. Ce dernier a été introduit en 2016, par R. Joseph. "You Only Look Once" (YOLO) adopte alors sa propre philosophie de détection : Le réseau divise l'image en plusieurs régions et prédit les Bounding Boxes et les probabilités de classes simultanément. YOLO est extrêmement rapide. Malgré sa grande amélioration de la vitesse de détection, YOLO souffre d'une baisse de la précision de localisation par rapport aux détecteurs à deux étages, en particulier pour certains petits objets. Les versions ultérieures de YOLO [50] [15]

[57] ont accordé plus d'attention à ce problème.

YOLOv3 : YOLOv3 [50] est une amélioration de la méthode YOLO proposée en 2018. C'est la dernière publication YOLO de Joseph Redmon [49] pour la détection d'objets. L'architecture de YOLOv3 est beaucoup plus grande que ses prédécesseurs avec 106 couches (Darknet-53 et 53 autres couches) et est entièrement convolutionnelle. Contrairement aux anciennes versions de YOLO, la détection d'objets ne se fait pas juste à la dernière couche, mais à 3 échelles différentes. Cela améliore la détection des petits et très grands objets. Récemment, YOLOv7 [57], un travail de suivi de l'équipe YOLOv4[15], a été proposé. Il surpasse la plupart des détecteurs d'objets existants en termes de vitesse et de précision.

DEtection TRansformer (DETR) : Ces dernières années, les transformateurs ont profondément influencé l'ensemble du domaine de l'apprentissage profond, en particulier le domaine de la vision par ordinateur. En 2020, N. Carion et al [18] ont proposé DETR, où ils ont considéré la détection d'objets comme un problème de prédiction d'ensemble et ont proposé un réseau de détection de bout en bout avec des transformateurs. Plus tard, X. Zhu et al [64] ont proposé Deformable DETR pour remédier au temps de convergence trop long de DETR et à ses performances limitées en matière de détection de petits objets. Il atteint des performances sur l'ensemble de données MSCOCO (COCO mAP@.5=71.9%).

Certes, la détection d'objets en général a fait des progrès significatifs, mais pour aborder la détection de météores, il est crucial de se pencher sur les défis et les avancées dans la détection de petits objets.

2.5 Détection de petits objets (DPO) :

2.5.1 Définition :

La détection de petits objets (small) ou d'objets minuscules (tiny) se concentre sur la détection d'objets de taille limitée. Les termes « minuscule » et « petit » sont généralement définis par un seuil de surface ou de longueur. Si l'on prend l'exemple du dataset MS COCO [40], les objets occupant une surface inférieure ou égale à 1024 pixels entrent dans la catégorie des objets de petite taille.

2.5.2 Défis de la DPO :

En plus de certains défis communs à la détection d'objets génériques, tels que les variations intra-classe, la localisation imprécise, la détection d'objets occultés, etc., des problèmes typiques existent dans la DPO, notamment la perte d'informations sur les objets, la représentation bruyante des caractéristiques, la faible tolérance à la perturbation de la boîte englobante et les échantillons inadéquats pour l'entraînement.

— **Perte d'informations :** Les détecteurs d'objets actuels utilisent généralement un extracteur de caractéristiques (backbone) et une tête de détection [51], [39] [15]. Cependant, les extracteurs de caractéristiques génériques [33], [59] ont tendance à utiliser

des opérations de sous-échantillonnage pour filtrer les activations bruyantes et réduire la résolution spatiale des cartes de caractéristiques, ce qui entraîne une perte significative d'informations pour les petits objets. Cette limitation compromet la capacité de la tête de détection à prédire avec précision les petits objets.

— **Représentation bruitée des caractéristiques :** Les caractéristiques sont essentielles pour les tâches de classification et de localisation. Cependant, les petits objets présentent souvent une faible résolution, ce qui rend difficile l'apprentissage de représentations à partir de leurs structures déformées. De plus, les caractéristiques des petits objets peuvent être polluées par le fond et d'autres instances, introduisant ainsi du bruit dans la représentation apprise.

— **Faible tolérance aux perturbations de la boîte englobante :** La localisation des objets est évaluée à l'aide de la métrique Intersection sur Union (IoU). Les petits objets montrent une sensibilité accrue aux variations de la boîte englobante prédite. Même de légères déviations peuvent entraîner une chute significative de l'IoU par rapport aux objets de taille moyenne et grande, rendant la régression de la boîte englobante plus complexe pour les petits objets.

2.5.3 Travaux sur la DPO :

Pour améliorer la détection des petits objets, plusieurs stratégies ont été mises en œuvre. Kisantal et al. [36] ont utilisé l'augmentation par copier-coller, copiant de petits objets avec des transformations aléatoires pour augmenter leur fréquence dans les images. Chen et al avec RRNet [21] ont appliqué AdaResampling, une méthode d'augmentation adaptative guidée par une carte de segmentation préalable pour positionner et redimensionner les objets collés afin de réduire les différences d'échelle. Zhang et al. [63] et Wang et al. [58] ont employé des techniques de division et de redimensionnement pour obtenir davantage d'échantillons d'entraînement de petits objets. Xu et al. [60] ont introduit la distance euclidienne normalisée DotD pour remplacer l'IoU, améliorant ainsi la correspondance des petits objets. En ce qui concerne les méthodes tenant compte de l'échelle, Yang et al. [61] ont développé la mise en commun dépendante de l'échelle (SDP) pour sélectionner les couches de caractéristiques adaptées aux petits objets. MS-CNN [17] génère des propositions d'objets à différentes couches en se concentrant sur des gammes d'échelles spécifiques. YOLOv3 [50] ajoute des branches parallèles pour des prédictions multi-échelles, avec des caractéristiques haute résolution ciblant les petits objets. Enfin, le réseau pyramidal de caractéristiques (FPN) [39] distribue les instances à différents niveaux de pyramide en fonction de leur taille, assurant ainsi une représentation multi-échelle appropriée.

2.6 Contribution

Pour répondre aux limites et aux défis de la détection des météores mis en évidence précédemment, nous avons introduit plusieurs solutions et contributions innovantes :

1. **Création d'un ensemble de données personnalisé :**
   — Nous avons compilé un ensemble de données diverses provenant de sources et de réseaux multiples, garantissant des conditions environnementales et des emplace-

ments géographiques variés. Cette robustesse des données d'entraînement permet au modèle de mieux se généraliser dans différents scénarios, améliorant ainsi la précision de la détection.

2. **Architectures améliorés**
   — Nous avons exploré et mis en œuvre diverses architectures d'apprentissage profond, Cette analyse comparative permet d'identifier le modèle le plus efficace pour détecter les petits objets météoriques et les distinguer des faux positifs.
3. **Réduction des faux positifs :**
   — En intégrant la suppression du non-maximum (NMS) et les statistiques d'ancrage, nous avons réduit de manière significative l'occurrence des faux positifs, qui sont des problèmes courants dans les tâches de détection des météores.
4. **Techniques d'augmentation des données**
   — Mise en œuvre de diverses méthodes d'enrichissement des données, telles que les retournements horizontaux et verticaux et les rotations de 90°. Ces techniques ont permis d'accroître la robustesse du modèle et sa capacité à gérer les diverses apparitions de météores et les conditions environnementales.
5. **Déploiement :**
   — Afin de réduire le temps de calcul, nous avons optimiser les paramètres ainsi que les poids du modèle pour une inférence plus rapide.
6. **Vérification multi-site :**
   — Développement d'un système permettant de vérifier si le même météore est observé par plusieurs caméras tout ciel situées à des endroits différents, ce qui améliore la fiabilité de la détection grâce à la vérification croisée des observations.

2.7 Conclusion

En conclusion, la détection des météores à l'aide de caméras tout ciel est une tâche complexe qui nécessite des algorithmes sophistiqués et des ressources informatiques considérables. Bien que des progrès significatifs aient été réalisés ces dernières années, plusieurs limitations et défis subsistent. Il s'agit notamment des problèmes liés au bruit de fond, au flou de mouvement, aux conditions d'éclairage variables et à la nécessité de disposer de capacités de traitement précis. Pour relever ces défis, il faut développer des algorithmes plus robustes et des techniques de pré-traitement efficaces.

Les recherches futures devraient se concentrer sur la création d'algorithmes capables de mieux gérer ces complexités, ainsi que sur l'amélioration des capacités de traitement en temps réel afin de faciliter l'adoption généralisée des systèmes de détection des météores.

Dans le chapitre suivant, nous nous pencherons sur la phase de conception de notre projet. Nous y détaillerons le processus de construction de notre ensemble de données, la conception et l'entraînement de nos modèles, et le déploiement du modèle. Ce chapitre fournira un aperçu approfondi des méthodes et des technologies que nous avons utilisées pour surmonter les défis identifiés dans l'état de l'art, ouvrant la voie à un système de détection de météores plus robuste et plus efficace.

Chapitre 3

Conception

3.1 Introduction

Dans ce chapitre, nous nous penchons sur la conception et l'exécution de notre système de détection des météores. Le processus comprend plusieurs étapes, allant de la collecte et du pré-traitement des données à l'entraînement et à l'évaluation du modèle. Notre objectif est de créer un modèle robuste et précis capable de distinguer les météores des autres objets dans diverses conditions environnementales. Nous discuterons de l'ensemble du processus, y compris la préparation des ensembles de données, la sélection des modèles et l'application de techniques avancées telles que le flux optique pour améliorer la précision de la détection. Ce chapitre présente les composants clés et les méthodologies mises en œuvre dans le cadre de notre projet, ce qui permet de bien comprendre le développement du système.

3.2 Description et Étapes de travail

Étape 1 : Collection des données

Notre travail a commencé par la collecte de données. Sachant qu'il est important de disposer d'un ensemble de données diversifié et étendu, nous nous sommes tournés vers diverses sources d'observation des météores.

1. **Scraping Data :** Des scripts automatisés utilisant BeautifulSoup, certifi et requests ont été nos outils de prédilection. Ces scripts ont été méticuleusement conçus pour récupérer des vidéos d'observation de météores à partir de plusieurs référentiels en ligne. Notre source principale était le Swedish AllSky Meteor Network, connu pour sa riche collection de vidéos de météores.
2. **Sources de données :** Pour garantir la diversité de l'ensemble des données, nous avons élargi nos sources au réseau FRIPON et aux stations d'observation météorologique canadiennes. Cette variété était cruciale pour capturer différentes perspectives et apparitions de météores, qui aideraient plus tard à former un modèle plus généralisable.

Étape 2 : Pré-traitement

Avec les données brutes en main, notre prochaine étape a été de les prétraiter pour s'assurer qu'elles conviennent à l'apprentissage du modèle.

1. **Extraction des images :** À partir des vidéos collectées, nous avons extrait des images individuelles, générant ainsi un ensemble substantiel de données d'images. Cette étape a été cruciale car elle a permis de convertir nos données vidéo dans un format pouvant être annoté et introduit dans nos modèles.
2. **Nettoyage des données :** Les images extraites ont fait l'objet d'un processus de nettoyage rigoureux. En collaboration avec l'équipe du CRAAG, nous avons filtré les doublons et les faux positifs, en veillant à ce que seules les images pertinentes et de haute qualité soient conservées dans notre ensemble de données.

Étape 3 : Annotation

Avec un ensemble de données propre, nous sommes passés à la phase d'annotation, qui était essentielle pour entraîner notre modèle à distinguer les météores des non-météores.

1. **Annotation manuelle :** À l'aide de Roboflow, nous avons téléchargé nos images pour les annoter manuellement. Chaque objet dans les images sélectionnées a été soigneusement étiqueté avec des étiquettes "météores" ou "non-météores". Ce processus a consisté à identifier les météores et à marquer les objets souvent confondus avec des météores, tels que les avions, les nuages, les insectes, les étoiles et les lumières de la ville. Cette annotation méticuleuse était essentielle pour créer un ensemble de données d'entraînement fiable.

Étape 4 : Augmentation des données

Pour améliorer encore notre ensemble de données, nous avons appliqué diverses techniques d'augmentation des données, ce qui a permis d'accroître sa diversité et d'aider notre modèle à mieux se généraliser.

1. **Techniques appliquées :** Nous avons mis en œuvre plusieurs techniques d'augmentation, notamment le retournement horizontal et vertical et les rotations de 90° (dans le sens des aiguilles d'une montre, dans le sens inverse des aiguilles d'une montre). Ces techniques ont permis de simuler différentes orientations et apparences des météores, rendant ainsi notre ensemble de données plus robuste.

Étape 5 : Conception du modèle

Avec un ensemble de données bien préparé, nous nous sommes lancés dans la phase de développement du modèle, en explorant différentes architectures et en affinant notre approche.

1. **Faster R-CNN avec ResNet Backbone :** Nous avons commencé le développement de notre modèle en utilisant l'architecture Faster R-CNN avec un backbone ResNet. Faster R-CNN est connu pour sa robustesse dans les tâches de détection d'objets, et ResNet a fourni une forte capacité d'extraction de caractéristiques.
2. **Faster R-CNN avec Swin Transformer Backbone :** Afin de comparer différents modèles et d'améliorer les performances, nous avons testé le transformateur Swin en tant que backbone de l'architecture Faster R-CNN. La structure hiérarchique et le mécanisme d'auto-attention du Swin Transformer ont permis d'améliorer les capacités de détection, en particulier pour les objets petits et variables comme les météores.

3. **Regularization et Dropout :** Pour éviter le sur-apprentissage et améliorer la généralisation du modèle, nous avons incorporé des techniques de régularisation, y compris des couches d'exclusion. Ces méthodes ont permis de maintenir les performances du modèle sur des données inédites.

Étape 6 : Post-traitement

Les étapes de post-traitement sont cruciales pour affiner les résultats de la détection et garantir une grande précision.

— **Non-Maximum Suppression (NMS) :** Nous avons mis en œuvre la NMS pour réduire les détections redondantes, en veillant à ce que chaque météore ne soit détecté qu'une seule fois par image.

Étape 7 : Conception Multi-site

Reconnaissant l'importance de vérifier les détections de météores à partir de plusieurs points de vue, nous avons développé un système de vérification croisée des détections à partir de plusieurs caméras AllSky.

— **Multi-Camera Verification :** Ce système a été conçu pour confirmer qu'un météore détecté par une caméra a également été observé par d'autres caméras situées à des endroits différents mais au même moment. En comparant les temps de détection, nous avons pu confirmer l'observation du même météore.

3.3 Dataset (Collection et Préparation)

Le succès de tout projet d'apprentissage automatique dépend fortement de la qualité et de la quantité des données. Dans le contexte de notre projet, l'objectif était de détecter avec précision les météores et les non-météores, ce qui nécessitait un ensemble de données robuste. Ce sous-chapitre détaille le processus de collecte, d'annotation et d'enrichissement des données, en soulignant les mesures prises pour garantir la pertinence et la qualité de l'ensemble de données.

3.3.1 Processus de Collect des données

Le processus de collecte des données a constitué une phase critique de ce projet, impliquant le scrapping automatisé des sites web accessibles au public qui offrent les données au format vidéo sous forme de pages web séparées (l'ensemble de données suédoises) [55] ou de lignes de tableau (FRIPON)[28] ou quelques vidéos du site web de l'Université de l'Ouest de l'Ontario au Canada (SOMN) [44]. L'objectif principal était de rassembler un ensemble varié de vidéos contenant des météores et des non-météores.

3.3.1.1 Swedish AllSky Meteor Network

Ce réseau compte sept stations actives. Le réseau repose en grande partie sur des stations peu coûteuses gérées par des particuliers ou de petites sociétés d'astronomes amateurs. Basé à l'origine sur le réseau météorologique danois Stjerneskud, le nœud central de l'université d'Uppsala fournit au réseau l'infrastructure nécessaire, comme la distribution d'un logiciel continuellement mis à jour et le traitement automatique des données provenant de toutes les

stations. Bien qu'il soit difficile de couvrir une très grande masse terrestre avec des ressources relativement faibles, plusieurs événements ont été bien observés jusqu'à présent, souvent en collaboration avec des observations provenant de pays voisins [53].

3.3.1.2 Canadian Dataset

SOMN Fireball Event : Sept caméras tout ciel du Southern Ontario Meteor Network (SOMN) de l'Université Western Ontario ont enregistré une boule de feu brillante dans le ciel du soir au-dessus de l'extrémité ouest du lac Ontario. La boule de feu a été vue par de nombreux observateurs dans tout le sud de l'Ontario et les régions adjacentes. La boule de feu a été détectée pour la première fois par les systèmes de caméras de l'Université Western à une altitude de 100 km au-dessus de Guelph, se déplaçant vers le sud-est à une vitesse de 20,8 km/s. Le météoroïde avait initialement la taille d'un tricycle d'enfant. Au plus fort de sa luminosité, la boule de feu était environ 100 fois plus brillante que la pleine lune. Ces vidéos provenant de différentes stations ont été incluses dans notre ensemble de données afin de permettre au modèle de s'entraîner sur une variété de perspectives[44].

3.3.1.3 Techniques de Web Scraping

Pour collecter des vidéos sur le web, nous avons employé des techniques de scraping web en utilisant des bibliothèques Python telles que BeautifulSoup et requests. Ces outils, utilisés dans notre script d'automatisation, nous ont permis d'extraire systématiquement des vidéos de divers sites web et forums d'astronomie.

3.3.1.4 Swedish Dataset l'ensemble de base

Ces vidéos détiennent la clé qui nous permettra d'obtenir des informations pour notre modèle de detection de météores. Nous allons nous plonger dans chaque image, en les extrayant et en les préparant méticuleusement pour créer un riche ensemble de données. Cet ensemble de données servira ensuite de base à l'entraînement de notre IA, lui permettant d'apprendre et de faire des prédictions précises.

3.3.1.5 Extraction des frames :

Pour s'assurer que notre modèle peut s'adapter à différents cas de météores et gérer des environnements bruyants, nous avons entrepris un processus méticuleux de sélection de vidéos pertinentes à partir de notre ensemble de données. L'objectif était de capturer un large éventail d'apparences et de comportements de météores, ainsi que diverses conditions environnementales, y compris différents scénarios d'éclairage et de météo. Ce processus de sélection était essentiel pour l'entraînement d'un modèle robuste capable de détecter avec précision les météores dans diverses conditions.

Une fois les vidéos pertinentes sélectionnées, nous avons extrait des images de ces vidéos pour créer un ensemble complet de données d'images. Nous avons utilisé un script personnalisé pour automatiser le processus d'extraction des images, en veillant à capturer des images à intervalles réguliers pour couvrir la durée totale de chaque vidéo. Nous avons ainsi obtenu

un ensemble varié d'images comprenant différentes phases de l'activité météorique.

3.3.1.6 Nettoyages des images :

Après l'extraction des images, l'étape suivante a consisté à les nettoyer et à les filtrer. Il s'agit d'éliminer les doublons et les faux positifs afin de garantir la précision et la fiabilité de notre ensemble de données. Le processus de nettoyage a été mené manuellement en collaboration avec l'équipe du CRAAG (Centre de Recherche en Astronomie, Astrophysique et Géophysique), qui a mis à disposition son expertise pour identifier les cas de météorites authentiques et différencier les faux positifs. Il s'agit des cas où l'image peut contenir des objets ou des phénomènes qui pourraient être identifiés par erreur comme des météores par le modèle. En éliminant ces faux positifs, nous avons amélioré la précision de notre ensemble de données.

3.3.2 Processus d'annotation

L'annotation précise des données était essentielle pour former un modèle fiable. Il s'agit d'étiqueter chaque image avec des boîtes englobantes et des étiquettes de classe (météore ou non-météore).

Dans notre projet, le processus d'annotation est une étape cruciale pour garantir la précision et l'efficacité de notre modèle de détection des météores. Il s'agit d'étiqueter manuellement les images extraites des vidéos afin de créer un ensemble de données bien annoté où chaque image possède des boîtes englobantes et des étiquettes de classe (météore ou non-météore). Voici une explication détaillée de la manière dont nous avons géré ce processus :

3.3.2.1 Upload vers Roboflow

Nous avons utilisé Roboflow, un outil d'annotation populaire, pour faciliter le processus d'annotation manuelle. Roboflow offre une interface intuitive pour l'étiquetage des images.

1. **Upload des images** :
   - Tout d'abord, nous avons uploadé les images extraites sur Roboflow.
   - Roboflow prend en charge divers formats d'images et offre un processus de téléchargement fluide, facilitant la gestion de grands ensembles de données.
2. **Création des étiquettes d'annotation** :
   - Sur Roboflow, nous avons créé deux étiquettes d'annotation : `meteor` et `nonmeteor`.
   - L'étiquette `meteor` a été attribuée à toute image contenant un météore.
   - L'étiquette `nonmeteor` a été utilisée pour les objets qui pourraient être confondus avec des météores.
3. **Annotation manuelle** :
   - Chaque image a été revue et annotée manuellement à l'aide des outils d'annotation de Roboflow.
   - Nous avons utilisé des boîtes englobantes pour marquer les régions d'intérêt dans chaque image.

- Les images contenant des météores ont été étiquetées avec le tag `meteor`, tandis que les objets pouvant être des faux positifs ont été étiquetés avec le tag `nonmeteor`.
- Cette annotation manuelle était essentielle pour garantir que notre modèle puisse distinguer avec précision les météores des autres objets, réduisant ainsi les risques de faux positifs.

4. **Contrôle de la qualité** :
   - Pour garantir l'exactitude des annotations, les images annotées ont été revues par le CRAAG.
   - Toute divergence dans les annotations a été discutée et résolue afin de maintenir un ensemble de données de haute qualité.
   - Ce processus de révision était crucial pour éliminer tout biais ou erreur potentiel dans les annotations.

3.3.2.2 Utilisation d'une annotation personnalisée

La décision d'utiliser à la fois les étiquettes `meteor` et `nonmeteor` a été guidée par notre analyse des ensembles de données existants, que nous avons examinés dans le deuxième chapitre de notre mémoire. Ces ensembles de données existants comportaient souvent des erreurs de classification, des objets tels que des avions, des nuages, des insectes, des étoiles et des lumières de ville étant considérés par erreur comme des météores. Ces erreurs de classification entraînaient un taux élevé de faux positifs, ce qui compromettait la précision des modèles de détection des météores.

En incorporant l'étiquette `nonmeteor`, nous avons cherché à entraîner notre modèle à reconnaître et à différencier les vrais météores des autres objets qui sont généralement pris pour des météores. Cette approche a été conçue pour combler les lacunes des ensembles de données existants et améliorer la fiabilité de notre modèle.

3.3.3 Techniques d'augmentation de données

Pour améliorer encore la robustesse de notre modèle, nous avons appliqué plusieurs techniques d'augmentation des données. Ces techniques aident le modèle à mieux se généraliser en introduisant des variations dans les données d'apprentissage. Les méthodes d'augmentation suivantes ont été utilisées :

- **Flip :** Retournements horizontaux et verticaux pour introduire des variations d'orientation.
- **Rotation 90° :** Rotation dans le sens des aiguilles d'une montre, dans le sens inverse des aiguilles d'une montre pour simuler différents points de vue.

En incorporant ces techniques d'augmentation, nous avons cherché à créer un ensemble d'entraînement plus diversifié qui pourrait améliorer la capacité du modèle à détecter les météores dans diverses conditions.

Impacte sur la performance : L'augmentation des données a considérablement amélioré la capacité de généralisation du modèle en l'exposant à une plus grande variété de conditions et de transformations d'images, améliorant ainsi ses performances sur des données inédites.

3.3.4 Statistiques

Pour tester véritablement la robustesse de notre modèle, nous avons créé deux versions de notre ensemble de données, en y incorporant un élément unique : les données du réseau canadien SOMN. Vous vous demandez peut-être pourquoi ce réseau en particulier ? La réponse réside dans la polyvalence que nous voulions obtenir en utilisant les données canadiennes, qui provenaient d'un ancien modèle de caméras datant de 2009. Ces caméras produisent des images de moindre résolution, ce qui permet à notre modèle de mieux se généraliser et d'apprendre davantage.

Nous avons notamment inclus quelques vidéos spectaculaires dans lesquelles le réseau a enregistré une boule de feu brillante à son maximum - près de 100 fois plus brillante que la pleine lune. En entraînant le modèle sur ces vidéos provenant de différentes stations, nous nous sommes assurés qu'il pouvait apprendre à partir d'une variété de perspectives, améliorant ainsi sa capacité à détecter et à classer les météores dans diverses conditions.

1. **Dataset V1 (Sans images Canadiennes) :**
   — Avant augmentation : 2037 image
   — Après augmentation : 5321 image

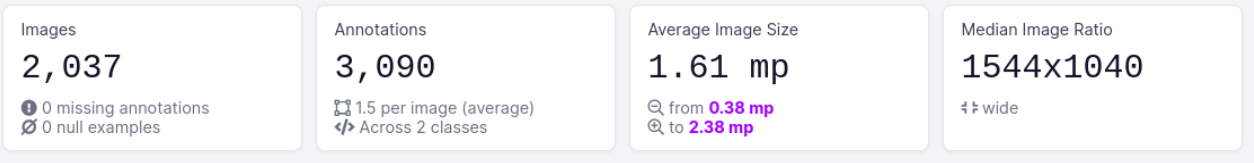

FIGURE 3.1 – Statistiques générales sur l'ensemble de données V1 sans les données canadiennes avant l'augmentation.

La figure 3.2 illustre la distribution des boîtes englobantes pour les météores et les non-météores dans la première version de notre ensemble de données, qui exclut les images canadiennes. Les zones vertes mettent en évidence les régions où les objets spécifiés apparaissent plus fréquemment, tandis que les zones violettes indiquent des apparitions moins fréquentes dans l'ensemble de la base de données. Il est intéressant de noter que les objets non météoriques ont des boîtes englobantes plus grandes, principalement en raison de la taille de la lune, qui est classée comme un objet non météorique. Cette visualisation donne un aperçu clair et perspicace de la distribution spatiale et des variations de taille entre les objets météoriques et non météoriques dans notre ensemble de données.

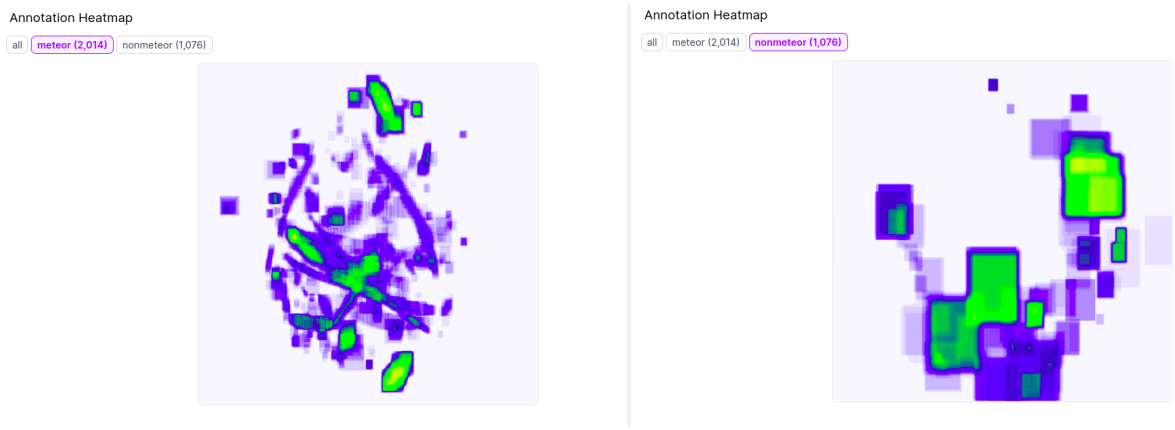

FIGURE 3.2 – Distribution des Apparitions de Météores (gauche) et Non-Météores (droite) dans des Images du Jeu de Données V1 Sans les Données Canadiennes

2. **Dataset V2 (Avec images Canadiennes) :**
   — Avant augmentation : 2160 image
   — Après augmentation : 5651 image

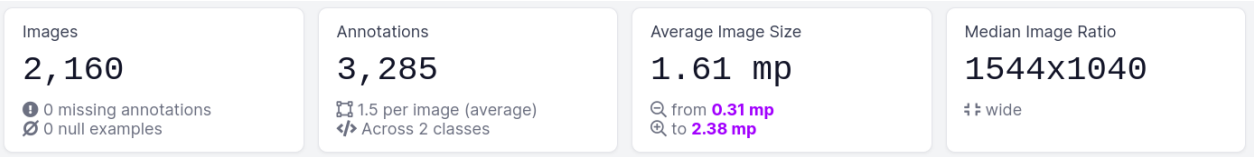

FIGURE 3.3 – Statistiques Générales sur le Jeu de Données V2 Avec les Données Canadiennes Avant l'Augmentation

La Figure 3.4 montre la distribution des boîtes englobantes pour les météores et les non-météores dans la deuxième version de notre ensemble de données, désormais enrichie d'images canadiennes. Les zones vertes indiquent les régions où les objets sont apparus plus fréquemment, tandis que les zones violettes indiquent des apparitions moins fréquentes. Cette version révèle notamment de nouvelles occurrences de boîtes englobantes dans des zones non couvertes par la première version de l'ensemble de données, ce qui met en évidence la variété et la généralité accrues des données. Cette comparaison souligne à quel point l'intégration des données canadiennes enrichit l'ensemble de données, en offrant une expérience d'entraînement plus diversifiée et plus robuste à notre modèle.

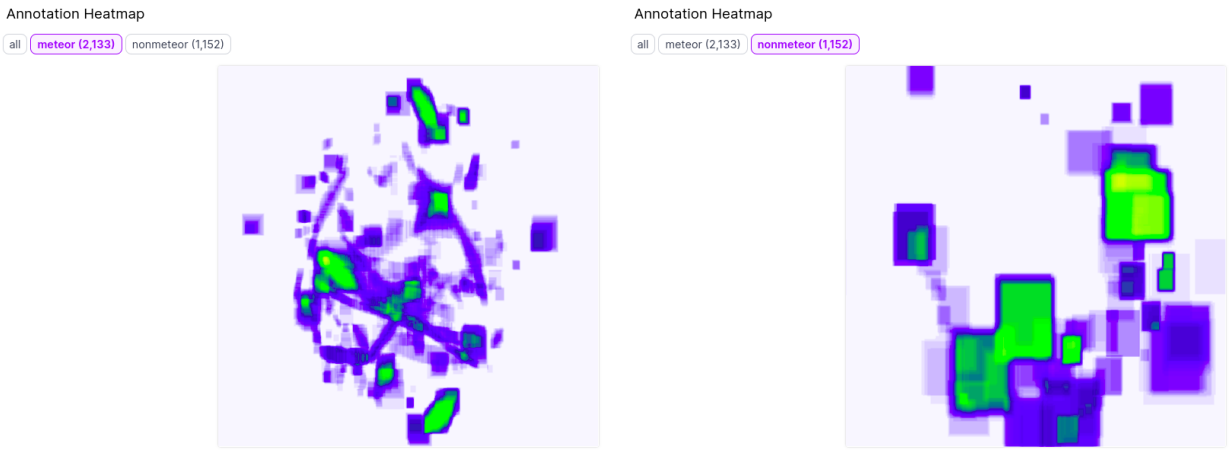

FIGURE 3.4 – Distribution des Apparitions de Météores (gauche) et Non-Météores (droite) dans des Images du Jeu de Données V2 avec les Données Canadiennes

3.3.5 Dataset Split

Pour garantir un ensemble de données d'entraînement et de validation équilibré et représentatif, nous avons divisé les données tout en maintenant un équilibre entre les différentes sources. Les sources sont les suivantes :

- **FRIPON**
- **Swedish**
- **Canadian**

La répartition a été effectuée de manière à ce que chaque sous-ensemble (entraînement et validation) contienne des représentations proportionnelles de chaque source. Cette stratégie a permis d'exposer le modèle à une variété d'observations météorologiques provenant de différents lieux géographiques et de différents réglages de caméra, ce qui est essentiel pour développer un modèle généralisable.

Nous nous sommes assurés que l'ensemble d'entraînement comportait des exemples diversifiés et que l'ensemble de validation fournissait une mesure fiable des performances du modèle sur différents types de données. Cette méthode a permis d'atténuer le risque que le modèle soit biaisé en faveur d'une source de données particulière.

La phase de collecte et de préparation des données a été complète et méthodique, garantissant que l'ensemble des données utilisées pour l'entraînement était de haute qualité et diversifié. Cette base était cruciale pour atteindre l'objectif du projet, à savoir la détection précise des météores et des non-météores. La combinaison de la collecte automatisée de données, de l'annotation méticuleuse et de l'augmentation efficace des données a joué un rôle essentiel dans la réussite du projet.

3.4 Conception des architectures de détection

Dans la section précédente, nous avons détaillé le processus méticuleux de collecte et d'annotation des ensembles de données. Sur cette base, nous nous concentrons maintenant sur la sélection d'un détecteur d'objets approprié, crucial pour obtenir des résultats précis et fiables, en particulier compte tenu des petits objets (météores) que nous cherchons à détecter. La Figure 3.5 illustre le pipeline de détection.

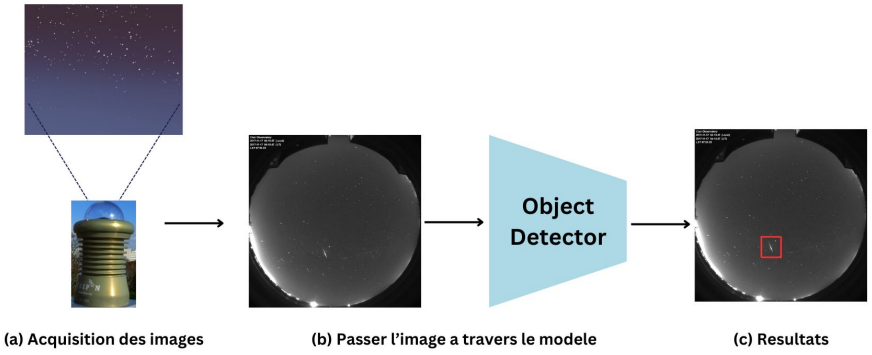

FIGURE 3.5 – Pipeline de détection.

3.4.1 Conception du détecteur de base (baseline de comparaison)

Un bon détecteur de base à choisir serait un détecteur en deux étapes, réputé pour sa précision et sa robustesse dans la détection d'objets de différentes tailles, Faster R-CNN. Une architecture de Faster R-CNN, se compose principalement (comme tout autre détecteur typique) de trois parties essentielles : le *backbone* (réseau dorsal), le *neck* (cou) et le *head* (tête de détection).

1. **Le Backbone :** Le backbone est responsable de l'extraction des caractéristiques de l'image. Il s'agit généralement d'un réseau de neurones convolutifs profonds, tel que ResNet ou VGG, qui a été préalablement entraîné sur un large ensemble de données (comme ImageNet). Le rôle du backbone est de transformer l'image d'entrée en une représentation riche de caractéristiques, capturant les détails nécessaires pour distinguer les objets. Dans le cas de la détection de petits objets comme les météores, un backbone avec une haute résolution de caractéristiques est crucial pour maintenir les détails fins nécessaires.
2. **Le Neck :** Le neck est une couche intermédiaire qui aide à renforcer et à fusionner les caractéristiques extraites par le backbone. Une des structures couramment utilisées pour le neck est le Feature Pyramid Network (FPN), qui construit des pyramides de

caractéristiques permettant de détecter des objets à différentes échelles. Cette capacité est particulièrement utile pour la détection de petits objets, car elle permet au modèle de conserver des informations de résolution fine tout en intégrant des informations de contexte globales.

3. **Le Head :** Le head est chargé de la classification et de la localisation des objets dans l'image. Dans une architecture en deux étapes, comme Faster R-CNN, le head est divisé en deux sous-modules : le RPN (Region Proposal Network) et le RoI (Region of Interest) head. Le RPN génère des propositions de régions d'intérêt, et le RoI head affine ces propositions pour prédire les classes et les boîtes englobantes des objets. Cette décomposition en deux étapes permet d'abord de filtrer les régions pertinentes avant d'affiner la détection, ce qui améliore la précision.
   — **RPN (Réseau de propositions de régions) :** Le RPN est un réseau entièrement convolutionnel qui prédit les limites de l'objet en apprenant à partir de cartes de caractéristiques extraites d'un réseau de base. Il comporte un classificateur qui renvoie la probabilité de la région et un régresseur qui renvoie les coordonnées des boîtes d'ancrage. 1000 propositions de boîtes (par défaut) avec des scores de confiance sont obtenues. La Figure 3.6 montre le fonctionnement du réseau de propositions.

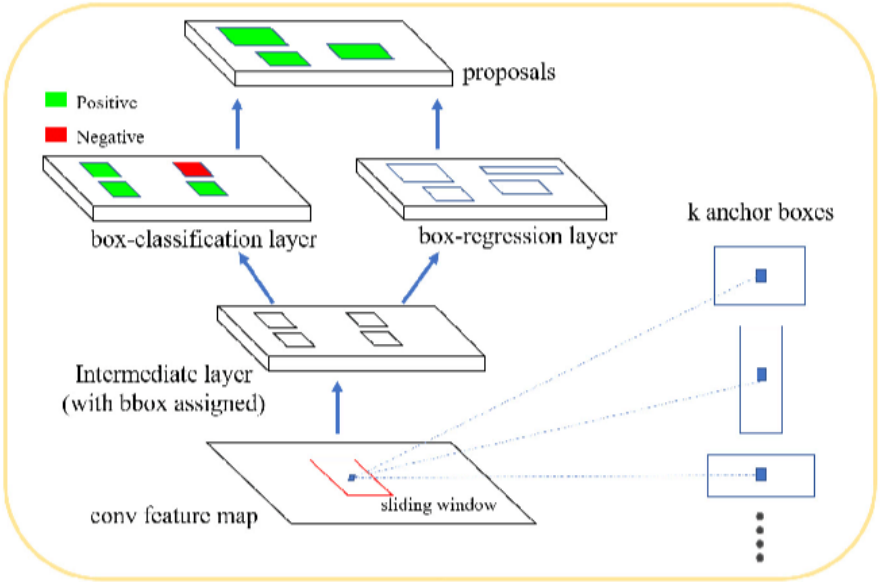

FIGURE 3.6 – RPN. [8]

   — **Boîtes d'ancrage :** Dans un réseau de neurones convolutif, une carte de caractéristiques est générée, et chaque point de cette carte, appelé ancre, est crucial pour la détection d'objets. Les boîtes d'ancrage sont des rectangles prédéfinis basés sur les rapports d'aspect et les échelles des images. Ces boîtes sont centrées sur chaque point d'ancrage de la carte de caractéristiques, et se déplacent sur toute la carte comme une fenêtre coulissante. Pour chaque position sur la carte de caractéristiques, si nous disposons de k boîtes d'ancrage, nous obtenons k prédictions. Chaque prédiction détermine si la boîte d'ancrage contient un objet (premier

plan) ou non (arrière-plan), via des classifications binaires. La Figure 3.7 montre le fonctionnement des boîtes d'ancrage.

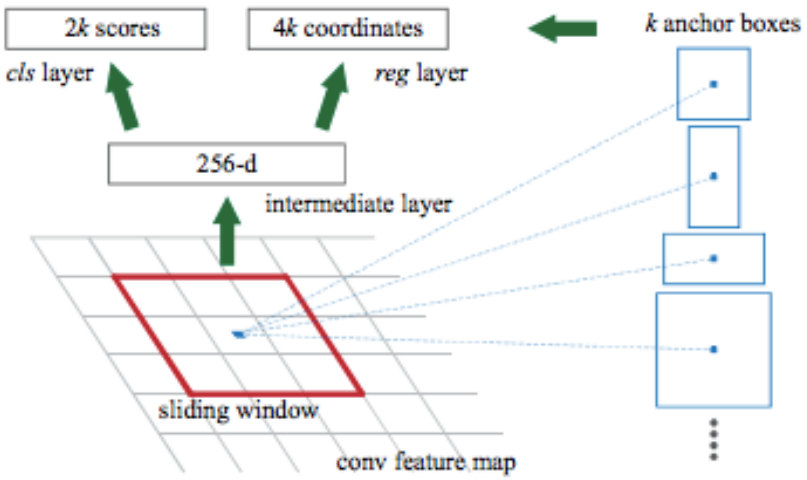

FIGURE 3.7 – Boîtes d'ancrages. [8]

— **ROI head (tête de boîte) :** Dans une architecture de détecteur en deux étapes comme Faster R-CNN, le RoI (Region of Interest) head joue un rôle crucial dans la phase finale de la détection des objets. Voici ce que fait le RoI head :

(a) Extraction des RoI : Le RoI head prend en entrée les régions proposées par le Region Proposal Network (RPN). Ces régions sont des propositions d'endroits dans l'image où des objets sont susceptibles de se trouver.

(b) Alignement des RoI : Pour traiter les propositions de régions d'intérêt de tailles variées, le RoI head utilise une opération appelée RoI Align.
   — **Interpolation Bilinéaire :** RoI Align utilise l'interpolation bilinéaire pour calculer les valeurs des pixels des caractéristiques aux positions continues, sans arrondir les coordonnées. Cela permet d'obtenir des valeurs de caractéristiques plus précises.
   — **Normalisation des RoI :** Les régions d'intérêt sont ensuite redimensionnées à une taille fixe, mais sans perdre les détails fins grâce à l'interpolation. En éliminant les erreurs d'alignement, RoI Align améliore la précision de la localisation des objets. Cela est particulièrement bénéfique pour la détection de petits objets comme les météores, où chaque pixel compte.

(c) Classification des RoI : Après l'alignement des RoI, le RoI head passe ces régions à travers des couches de réseaux de neurones (des couches entièrement connectées) pour classifier chaque région. Cela signifie qu'il détermine si la région contient un objet et, le cas échéant, de quelle classe il s'agit (par exemple, un météore dans notre cas).

(d) Régression des Boîtes Englobantes : En parallèle de la classification, le RoI head effectue également une régression pour ajuster les coordonnées des boîtes englobantes proposées par le RPN. Cela affine la localisation des objets dans

l'image, en ajustant les boîtes pour qu'elles correspondent plus précisément aux contours des objets détectés.

La Figure 3.8 illustre l'architecture de ROI Head de Fast RCNN.

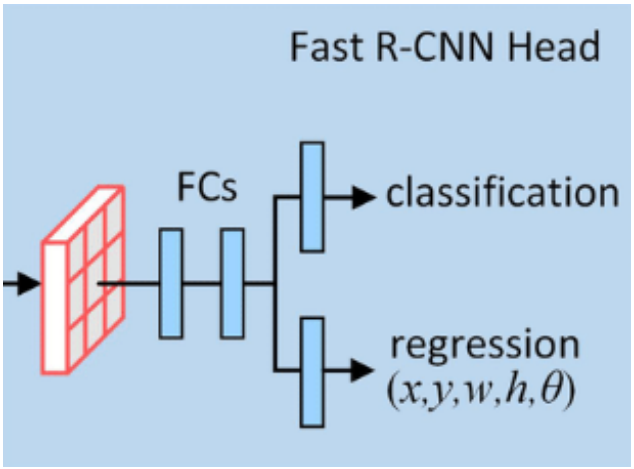

FIGURE 3.8 – ROI head [22]

3.4.1.1 Établir les éléments du baseline

Dans ce qui suit, nous construirons l'architecture Faster RCNN la plus adaptée à notre tâche spécifique de détection de météores, en choisissant soigneusement ses composants qui sont :

— **Le backbone**
— **Le neck**

Cela nous permettra d'optimiser le détecteur pour obtenir un détecteur robuste et solide comme base de comparaison dans la détection des petits objets dans nos données.

1. **Choix du backbone :**
   Dans notre quête d'optimisation de Faster R-CNN, nous avons choisi ResNet-50 comme backbone. ResNet-50, une variante des réseaux résiduels (Residual Networks), a été largement adopté en raison de sa capacité à atténuer le problème de disparition du gradient (vanishing gradient) grâce à l'introduction de connexions résiduelles. La Figure 3.9 montre l'architecture d'un backbone ResNet50.

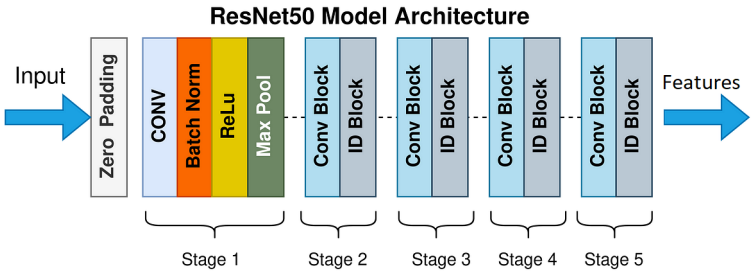

FIGURE 3.9 – Architecture de ResNet50.

Les raisons de ce choix sont :

— **Efficacité :** Par rapport à ResNet-101, ResNet-50 est moins gourmand en ressources informatiques, ce qui se traduit par des temps de training plus courts et un meilleur temps d'inférence.

— **Extensibilité :** La conception de ResNet-50 permet d'intégrer facilement des couches et des composants supplémentaires, tels que le Feature Pyramid Network (FPN), afin d'améliorer encore ses capacités.

— **Des performances dans l'état de l'art :** ResNet-50 offre toujours une grande précision lorsqu'il est utilisé avec Faster R-CNN. ResNet-50 a été largement testé et validé dans le cadre de divers défis et tests de référence en matière de détection d'objets, ce qui constitue une base solide pour notre modèle.

2. **Choix du neck :**
Le réseau pyramidal de caractéristiques (Feature Pyramid Network FPN) est un composant crucial dans de nombreuses architectures de détection d'objets, y compris Faster R-CNN. FPN améliore les capacités de détection multi-échelles du modèle en construisant une pyramide de caractéristiques à plusieurs niveaux, améliorant ainsi la précision et la robustesse pour différentes tailles d'objets. Nous choisissons donc FPN comme neck pour notre modèle de base.
Architecture FPN :
FPN (voir Figure 3.10) consiste de plusieurs composants clés :

— **1. Extraction de caractéristiques :** Le backbone (dans notre cas, ResNet50) extrait les caractéristiques de l'image d'entrée à plusieurs échelles. Ces caractéristiques sont ensuite introduites dans le FPN.

— **2. Chemin descendant (Top-Down) :** La voie descendante consiste à suréchantillonner les caractéristiques des échelles supérieures pour les faire correspondre à la résolution des échelles inférieures. Cette opération est réalisée à l'aide d'une série de couches convolutives avec un pas de 2.

— **3. Connexions latérales (lateral connections) :** Les connexions latérales relient les caractéristiques du chemin descendant aux caractéristiques correspondantes du chemin ascendant. Cela permet de combiner efficacement les caractéristiques provenant de différentes échelles.

— **4. Chemin ascendant (Bottom-up) :** La voie ascendante implique un sous-échantillonnage des caractéristiques des échelles inférieures pour correspondre à la

résolution des échelles supérieures. Cette opération est réalisée à l'aide d'une série de couches convolutives avec un pas de 2.

— **5. Fusion des caractéristiques :** Les caractéristiques des voies descendante et ascendante sont fusionnées à l'aide d'une addition par éléments. Ce processus de fusion permet de combiner les forces des caractéristiques provenant de différentes échelles.

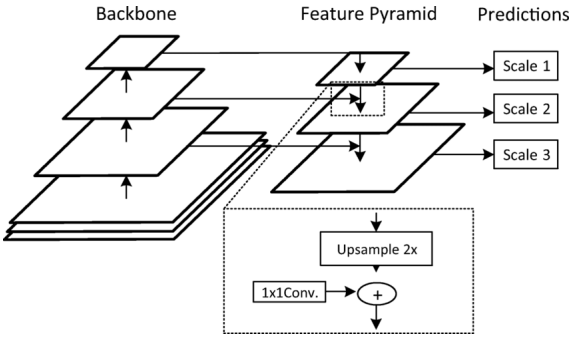

FIGURE 3.10 – Architecture de FPN [20].

La Figure 3.11 illustre en détail notre architecture Resnet-Faster-RCNN avec FPN.

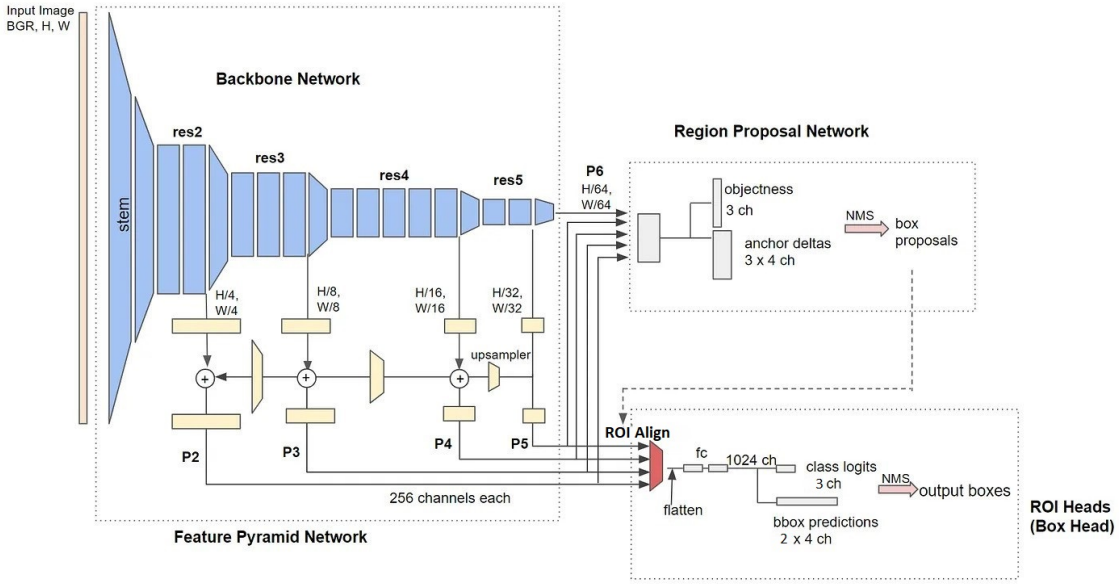

FIGURE 3.11 – Architecture du détecteur de base Faster R-CNN + ResNet-50 + FPN [1].

3.4.2 Conception d'un détecteur Amélioré

Après avoir établi une base solide, notre prochaine étape consiste à explorer les améliorations susceptibles d'accroître les performances de notre cadre de détection d'objets.

3.4.2.1 Modification du backbone

Notre approche consiste à intégrer Swin Transformer tiny comme nouveau backbone. Swin Transformer a démontré des capacités exceptionnelles dans diverses tâches de vision, ce qui en fait un candidat idéal pour l'amélioration de l'extraction des caractéristiques.

1. **Architecture de Swin Transformer :**
   L'architecture du Swin Transformer est conçue pour gérer de grandes images en les divisant en patches non superposés et en traitant ces patches à l'aide d'une approche hiérarchique. Cette approche permet au modèle de maintenir un temps de calcul raisonnable tout en traitant efficacement de grandes images. La Figure 3.12 illustre l'architecture de Swin Transformer-tiny, une version plus petite du Swin Transformer de base.

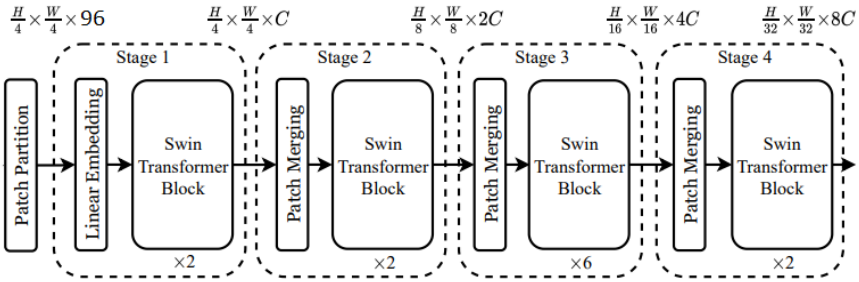

FIGURE 3.12 – Architecture de Swin Transformer-tiny [41].

— **Patch Splitting (Fractionnement des patches) :** L'image d'entrée est divisée en patches non superposés. Chaque patch est traité comme un "token" et sa caractéristique est définie comme une concaténation des valeurs brutes des pixels RGB.
— **Linear Embedding (Intégration linéaire) :** Une couche d'intégration linéaire est appliquée à la caractéristique en valeurs brutes pour la projeter dans une dimension arbitraire (dénotée par C).
— **Swin Transformer Blocks (Blocs de transformers SWIN) :** Plusieurs blocs Transformer avec un calcul de self-attention (auto-attention) modifié sont appliqués à ces tokens de patch. Ces blocs maintiennent le nombre de tokens et sont appelés "Stage 1".
— **Patch Merging (Fusion des patches) :** Le nombre de tokens est réduit par des couches de fusion de patches à mesure que le réseau devient plus profond. La première couche de fusion de patches concatène les caractéristiques de chaque groupe de 2×2 patches voisins et applique une couche linéaire sur les caractéristiques concaténées 4C-dimensionnelles. Cela réduit le nombre de tokens par un multiple de $2 \times 2 = 4$ (réduction de la résolution de 2 fois), et la dimension de sortie est fixée à 2C.
— **Répétition du Processus :** Cette procédure est répétée deux fois, avec le nombre de tokens réduit par une autre couche de fusion de patches à chaque itération. La

sortie de chaque étape est appelée "Stage 2," "Stage 3," et "Stage 4".

2. **Composants de Swin Transformer Block :**
   L'architecture du Swin Transformer Block inclut deux composants clés :
   — **Window Multihead Self-Attention (W-MSA) :** Ce module divise la séquence d'entrée en plusieurs segments de taille fixe ou "fenêtres" et applique la self-attention à chaque fenêtre indépendamment. Cette approche aide à réduire le coût computationnel de la self-attention en permettant le traitement parallèle des fenêtres.
   — **Shifted-Window Multihead Self-Attention (SW-MSA) :** Ce module décale les fenêtres d'un certain montant (par exemple, la moitié de la taille de la fenêtre) pour chevaucher les fenêtres adjacentes. Cette approche aide à atténuer le problème de perte d'information tout en permettant le traitement parallèle.
   — Le bloc Swin Transformer comprend également une normalisation des couches (LayerNorm) et un perceptron multicouche (MLP) appliqué entre les couches W-MSA et SW-MSA comme l'indique la figure 3.13 qui illustre deux blocs consecutifs.

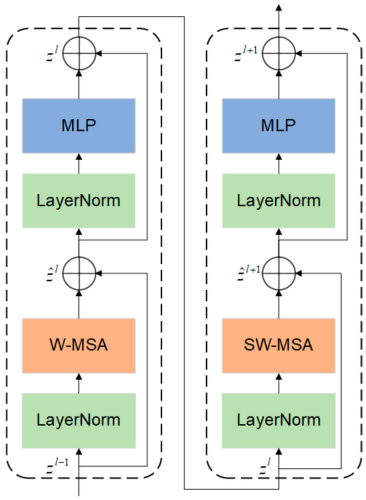

FIGURE 3.13 – Deux Blocs Swin Transformer consécutifs [41].

3. **Pourquoi Swin et non pas ViT (Vision Transformer) ?**
   Le transformateur Swin présente plusieurs avancées par rapport aux backbone conventionnelles traditionnelles et ViT par :
   — **Compréhension du contexte global :** Contrairement aux réseaux convolutifs traditionnels, le Swin Transformer capture les dépendances globales grâce à son mécanisme de self-attention. Cette compréhension du contexte global est particulièrement utile pour la détection d'objets petits et dispersés.
   — **Représentation hiérarchique :** Contrairement au ViT, qui traite les images à une seule échelle, Swin Transformer imite la structure hiérarchique des CNN, ce qui lui permet de traiter efficacement des caractéristiques à plusieurs échelles. La figure 3.14 montre les deux représentations, Swin et ViT côte à côte.

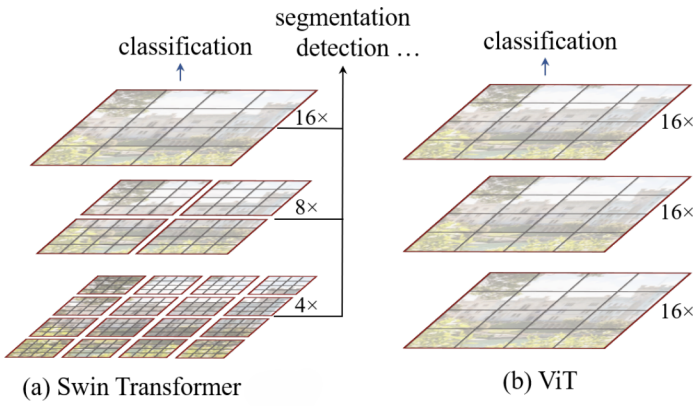

FIGURE 3.14 – Structure hiérarchique de Swin vs Structure de ViT [41].

— **Efficacité :** Swin Transformer introduit une approche de fenêtre décalée qui réduit considérablement la complexité de calcul par rapport à ViT. Il est donc plus efficace pour l'apprentissage et l'inférence, en particulier pour les grands ensembles de données ou les images à haute résolution.
— **Performances sur Benchmark :** Les résultats empiriques ont montré que le Swin Transformer est plus performant que ViT dans plusieurs tests de référence pour la détection d'objets, ce qui indique sa supériorité pour relever les défis de la détection dans le monde réel.

La Figure 3.15 illustre en détail l'architecture Swin-Faster-RCNN avec FPN obtenue avec le réseau Swin comme backbone.

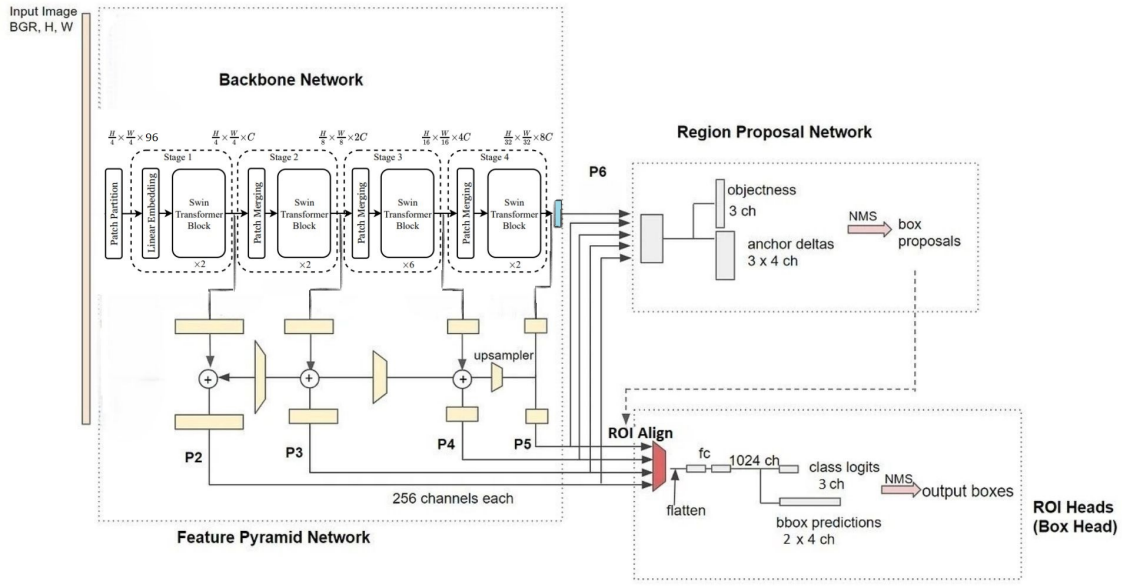

FIGURE 3.15 – Architecture du détecteur proposé Faster R-CNN + SWIN backbone + FPN.

3.4.2.2 Entrainement de l'architecture

1. Entraînement de bout en bout : Fine tuning l'ensemble du réseau Faster R-CNN, y compris le backbone, RPN et les têtes de détection, est ajusté de bout en bout à l'aide de la rétropropagation (voir Figure 3.16). Cette étape permet d'optimiser conjointement la génération de propositions de régions et la détection d'objets. Au départ, certaines couches du backbone peuvent être gelées pour conserver les caractéristiques apprises, en particulier si l'ensemble de données est petit.

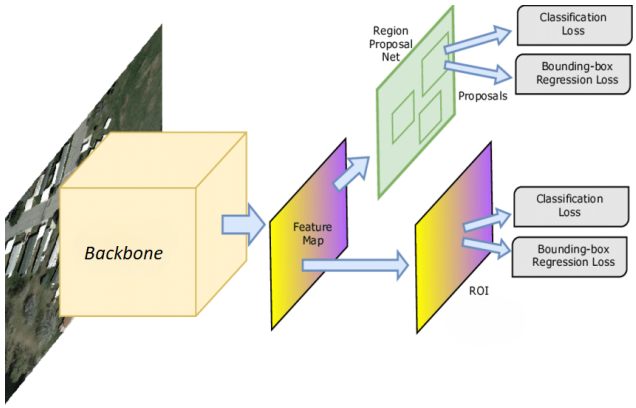

FIGURE 3.16 – Entrainement de Faster RCNN.

2. Perte combinée : calcule de la perte globale, qui comprend la perte de RPN et la perte de classification et la perte de régression de la boîte englobante.

— **Perte de RPN :**

$$L_{RPN}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{3.1}$$

Où :

— p_i : Probabilité prédite que l'ancre i soit un objet.
— p_i^* : Étiquette de vérité terrain (1 si l'ancre est positive, 0 si négative).
— t_i : Décalage de régression de la boîte englobante prédit pour l'ancre i.
— t_i^* : Décalage de régression de la boîte englobante de vérité terrain.
— N_{cls} : Nombre d'ancres dans le mini-lot (utilisé pour la normalisation).
— N_{reg} : Nombre d'ancres pour lesquelles la perte de régression est calculée (généralement le nombre d'ancres positives).
— λ : Paramètre de pondération pour équilibrer les deux termes de la perte.
— L_{cls} : Perte de classification (perte par entropie croisée).
— L_{reg} : Perte de régression (perte smooth L1).

— **Perte de ROI head :**

$$L_{ROI}(p_i, t_i) = \frac{1}{N} \sum_i [L_{cls}(p_i, p_i^*) + \lambda(p_i^* > 0) L_{reg}(t_i, t_i^*)] \tag{3.2}$$

Où :

— p_i : Distribution de probabilité prédite sur $K + 1$ classes (fond + K classes d'objets).
— p_i^* : Étiquette de classe de vérité terrain (encodée en one-hot).
— t_i : Décalages de régression de la boîte englobante prédits pour la classe p_i^*.
— t_i^* : Décalages de régression de la boîte englobante de vérité terrain.
— N : Nombre de régions d'intérêt (ROIs) dans le mini-lot (utilisé pour la normalisation).
— λ : Paramètre de pondération pour équilibrer les deux termes de la perte.
— L_{cls} : Perte de classification (perte par entropie croisée).
— L_{reg} : Perte de régression (perte smooth L1).

— **Perte totale :** La somme des deux pertes mentionnées ci-dessus est calculée pour mettre à jour les poids du modèle via la rétropropagation.

3. Amelioration du seuil de IoU :Pour l'entraînement de Faster RCNN, on trouve deux seuils de IoU, celui du RPN et celui de RoI head. En modifiant ces seuils, nous permettons au modèle d'apprendre à mieux détecter les petits objets. En particulier, ces ajustements réduisent les exigences pour qu'une boîte englobante soit considérée comme une correspondance positive, ce qui améliore la capacité du modèle à gérer les petites déviations de localisation et à maintenir une performance acceptable sur les petits objets. Cela permet d'augmenter la tolérance du modèle aux perturbations de la boîte englobante, un défi que nous avons rencontré dans l'état de l'art de la détection de petits objets, améliorant ainsi la précision de la détection des météores.

3.4.2.3 Améliorations Post-Entraînement

Appliquer NMS (Non Maximum Suppression) :
Non-Maximum Suppression (NMS) est une étape cruciale dans les algorithmes de détection d'objets, utilisée pour éliminer les détections redondantes d'un même objet. L'algorithme fonctionne en sélectionnant la détection avec le score de confiance le plus élevé et en supprimant toutes les autres détections qui ont une forte similarité avec elle, mesurée par le coefficient d'intersection sur union (IoU). L'IoU est le rapport entre l'intersection et l'union des boîtes englobantes des deux détections.

1. ***Pourquoi Réduire le Seuil d'IoU pour les Petits Objets ?***
   Lorsqu'on travaille avec des petits objets, réduire le seuil d'IoU est essentiel pour les raisons suivantes :
   — Proximité Spatiale : Les petits objets sont souvent proches les uns des autres dans l'image, et un seuil d'IoU élevé pourrait considérer plusieurs petites détections correctes comme redondantes et les supprimer, même si elles représentent des objets distincts.
   — Précision de la Détection : Les boîtes englobantes des petits objets peuvent ne pas être parfaitement alignées en raison de la résolution limitée des caractéristiques utilisées pour la détection. Un seuil d'IoU élevé pourrait donc injustement éliminer des détections correctes mais légèrement décalées. En réduisant le seuil d'IoU, nous

permettons à plus de détections proches mais distinctes d'être conservées, ce qui est crucial pour une détection précise des petits objets.

2. ***Influence de NMS sur la mAP (Mean Average Precision) :***
   La mAP (Mean Average Precision) est une métrique qui évalue la précision globale d'un modèle de détection en prenant en compte à la fois le rappel (recall) et la précision (precision). Voici comment NMS et le seuil d'IoU influencent la mAP :
   — Précision : Un seuil d'IoU bien ajusté garantit que les détections redondantes sont efficacement supprimées, ce qui améliore la précision. Cependant, un seuil trop bas pourrait éliminer trop de détections correctes, réduisant ainsi la précision.
   — Rappel : Si le seuil d'IoU est trop élevé, des détections valides pourraient être supprimées, diminuant le rappel. Pour les petits objets, un seuil plus bas aide à maintenir un bon rappel en conservant plus de détections proches.
   — Équilibre Précision-Rappel : La mAP est influencée par l'équilibre entre la précision et le rappel. Un seuil d'IoU réduit pour les petits objets aide à maintenir ce délicat équilibre en conservant suffisamment de détections pour un rappel élevé, tout en réduisant les faux positifs pour maintenir la précision.

3.4.2.4 Déploiement

Les modèles de Deep Learning sont souvent très "lourds" et nécessitent une puissance de calcul considérable, ce qui implique généralement l'utilisation de machines équipées de GPU pour gérer leur complexité. Cela limite leur déploiement sur des appareils disposant de ressources de calcul restreintes (CPU). De plus, CUDA toolkit nous oblige à n'utiliser que des appareils Nvidia.

Dans le cadre de ce projet, nous proposons une solution qui permettra à ces modèles d'être déployés efficacement sur n'importe quelle machine, qu'elle soit équipée d'un CPU ou d'un GPU. Cela vise à démocratiser l'accès aux capacités avancées du Deep Learning en éliminant les contraintes matérielles et en rendant la technologie plus accessible et plus flexible pour une variété d'applications.

Solution OpenVino :

Nous avons choisi d'utiliser la bibliothèque OpenVino, car elle nous permet d'exécuter le modèle sur un CPU ou un GPU Intel standard qui se déploie sur n'importe quelle machine, ainsi que sur des machines plus petites comme un raspberry PI + une unité de traitement de la vision (VPU) externe. Elle ne nécessite pas CUDA toolkit qui est obligatoire pour utiliser toute autre bibliothèque comme Pytorch, et nous permet de convertir les modèles depuis différentes librairies et de les optimiser (Comme on peut le voir sur la Figure 3.17)

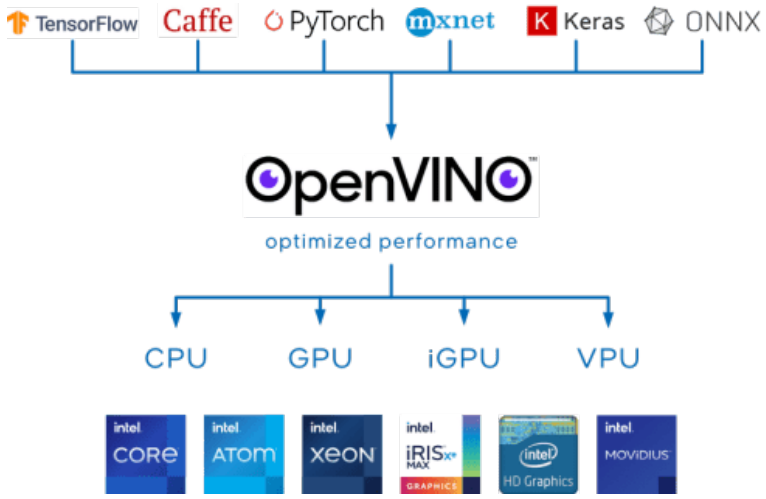

FIGURE 3.17 – Convertir de n'importe quelle bibliothèque vers OpenVino. [5]

L'un des avantages d'OpenVino est d'accélérer le temps d'inférence avec un faible compromis sur la précision, en utilisant différents types d'optimisations. Nous utiliserons l'optimisation post-entraînement, puisque nous avons entraîné notre modèle sur pytorch. Nous utiliserons la quantification FP16 en précision mixte.

Étapes du déploiement du modèle :

1. **Exporter vers ONNX :** Puisque nous travaillons avec pytorch, nous devons exporter le modèle de pytorch vers ONNX (Open Neural Network Exchange), qui est le format standard pour les modèles d'apprentissage profond.
2. **Exporter vers format OpenVino IR :** avec notre modèle ONNX nouvellement obtenu, nous pouvons maintenant utiliser le Model Optimizer MO de la bibliothèque OpenVino pour exporter vers le format OpenVino IR qui comprend :
   (a) Fichier XML pour l'architecture et les paramètres du modèle (taille d'entrée, couches... etc)
   (b) Fichier BIN pour les poids
3. Au cours du processus de conversion, MO nous permet d'activer la précision mixte FP16 pour une inférence plus rapide. Les étapes sont les suivantes :
   (a) Weights Conversion : Les poids stockés dans le fichier .BIN sont convertis de FP32 en FP16. La taille de stockage des poids est ainsi réduite, ce qui permet de réduire considérablement l'empreinte mémoire.
   (b) Transformation des types de données : Les types de données dans la structure du modèle définie dans le fichier .XML sont également mis à jour (de FP32 à FP16).
4. Maintenant que les fichiers XML et bin sont convertis en FP16, nous pouvons les déployer sur notre machine avec GPU Intel (figure 3.18).
5. En option, nous pouvons également quantifier le modèle à la précision INT8, ce qui peut augmenter la vitesse jusqu'à 3 fois de FP32 à INT8.

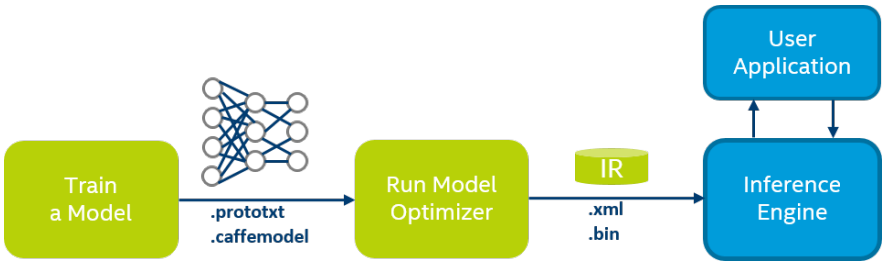

FIGURE 3.18 – Étapes pour inférer un modèle avec OpenVino [10]

3.5 Détection multi-site

Dans le contexte de la détection multi-sites des événements météoriques, notre objectif est de vérifier si deux caméras distinctes, placées à différents sites, ont détecté le même événement météorique. Cette vérification est essentielle pour valider et corréler les observations de météores sur plusieurs emplacements.

Pour ce faire, nous proposons un algorithme qui implique des fenêtres temporelles superposées. Chaque caméra génère une liste de moments d'apparition, indiquant quand un météore a été détecté dans son champ de vision. Ces moments d'apparition représentent essentiellement des intervalles de temps pendant lesquels le météore a été observé par chaque caméra.

L'approche consiste à vérifier les chevauchements entre ces intervalles de temps. S'il y a un chevauchement significatif entre les moments d'apparition des deux caméras (c'est-à-dire si les intervalles de temps des deux caméras se croisent selon un seuil spécifié), cela suggère que le même événement météorique a été observé par les deux caméras. Cette vérification aide à confirmer la cohérence et la fiabilité de la détection des météores sur différents sites.

Dans l'algorithme 1 si dessous, décrit ce processus de vérification. L'algorithme parcourt les moments d'apparition des détections météoriques provenant de deux vidéos différentes (***appearance_times1*** et ***appearance_times2***). Il vérifie s'il y a un chevauchement entre les intervalles de temps en appliquant une condition.

Algorithme 1 : Vérification du chevauchement

Entrée : Listes des temps d'apparitions des météores **appearanceTimes1**, **appearanceTimes2**, Réel : **seuil**
sortie : Booléen : **B**
Debut
$B \leftarrow$Faux
pour ***start1***, ***end1*** *in* ***appearanceTimes1*** **faire**
  pour ***start2***, ***end2*** *in* ***appearanceTimes2*** **faire**
    si *Maximum(**start1**,**start2**) < Minimum(**end1**,**end2**)* **alors**
      $Chevauchement \leftarrow Minimum(\textbf{end1},\textbf{end2}) - Maximum(\textbf{start1},\textbf{start2})$;
      si *Chevauchement > seuil* **alors**
        $B \leftarrow$Vrai;
        Retourner B
      fin
    fin
  fin
fin
Retourner B
Fin

Ici :

- ***start1*** et ***end1*** représentent le début et la fin de l'intervalle d'apparition de la première caméra respectivement.
- ***start2*** et ***end2*** représentent le début et la fin de l'intervalle d'apparition de la deuxième caméra respectivement.
- ***seuil*** est un paramètre définissant la durée minimale de chevauchement requise pour considérer les événements comme se chevauchant.
- Si la condition est remplie, indiquant chevauchement supérieur au seuil, la fonction renvoie ***True***, ce qui signifie que le même événement météorique a été détecté par les deux caméras. Sinon, elle renvoie ***False***, indiquant l'absence de chevauchement.

3.6 Conclusion

Notre travail vers le développement d'un système robuste de détection des météores a commencé par une approche méthodique. Chaque étape que nous avons entreprise dans ce chapitre a été une étape critique de notre conception pour la détection de météores. Nous avons commencé par une collecte de données provenant de sources diverses, assurant ainsi un ensemble de données riche et varié. Cela a été suivi par un prétraitement des données et une annotation manuelle pour fournir des données d'entraînement de haute qualité. Nous avons ensuite appliqué des techniques d'augmentation de données pour enrichir notre ensemble de données.

Le développement du modèle a été une phase critique au cours de laquelle nous avons exploré des architectures telles que Faster R-CNN avec des backbones ResNet et Swin Transformer, en incorporant des techniques de régularisation pour prévenir le sur-apprentissage.

Nous avons proposé des techniques avancées de détection, telles que Suppression Non Maximale (NMS) pour affiné les performances de notre modèle. Nous avons également mis en place un système de vérification multi-site.

En résumé, ce chapitre nous avons posé les bases solides de notre système de détection des météores, préparant le terrain pour les résultats expérimentaux présentés dans le chapitre suivant.

Chapitre 4

Résultats Expérimentaux

4.1 Introduction

Dans ce chapitre, nous nous penchons sur les aspects techniques et les résultats de notre projet de détection de météores. Nous décrivons l'environnement informatique, les outils et les bibliothèques utilisés, puis nous analysons en profondeur le processus d'entraînement et les résultats. Ce chapitre comprend également des évaluations quantitatives et qualitatives, un aperçu de l'interface graphique de l'utilisateur (GUI) et des détails sur le système de détection multi-événements. Notre objectif est de fournir une compréhension complète de notre mise en œuvre et des mesures de performance qui en résultent.

4.2 Environnements de Travail

En raison des limitations de nos machines personnelles, qui ne disposent pas des GPU NVIDIA essentiels à cette tâche, la phase de collecte et de préparation des données a été exécutée sur des machines personnelles locales. Pour la phase d'entraînement, nous avons utilisé de Kaggle Notebooks, en utilisant le GPU P100 pour un entraînement et une évaluation efficaces des modèles. L'environnement du système était le suivant :

- **Platform** : Linux
- **Python** : 3.10.13 (packaged by conda-forge)
- **CUDA** : Available (Version 12.1)
- **GPU** : Tesla P100-PCIE-16GB
- **CUDA_HOME** : /usr/local/cuda
- **NVCC** : Cuda compilation tools, release 12.1, V12.1.105
- **GCC** : gcc (Ubuntu 9.4.0-1ubuntu1 20.04.2) 9.4.0
- **PyTorch** : 2.1.2
- **TorchVision** : 0.16.2
- **OpenCV** : 4.9.0

PC1 :

- **Device name** : Dell Latitude E7470
- **Processor** : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz
- **Installed RAM** : 8.00 GB (7.89 GB usable)
- **Graphics** : Intel® UHD Graphics 520
- **Disk Capacity** : 256GB
- **System type** : 64-bit operating system, x64-based processor
- **Python** : 3.10.13

PC2 :

- **Device name** : Dell Inc. Inspiron 5570
- **Processor** : Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
- **Installed RAM** : 8.00 GB (7.89 GB usable)
- **Graphics** : Intel® UHD Graphics 620 (KBL GT2)
- **Disk Capacity** : 1.3TB
- **System type** : Pop!_OS 22.04 LTS 64-bit
- **Python** : 3.10.13

4.3 Environnement Logiciel

- **Python** : Il est utilisé pour le traitement des données, l'entraînement et l'évaluation des modèles. [7]
- **Jupyter Notebooks** : Utilisé pour l'organisation du code et l'exécution interactive. [35]
- **PyTorch** : Pour développer et entraîner des modèles d'apprentissage profond. PyTorch est une bibliothèque d'apprentissage automatique open-source. [11]
- **OpenCV** : Pour les tâches de traitement d'images. OpenCV (Open Source Computer Vision Library) est une bibliothèque libre pour la vision par ordinateur et l'apprentissage automatique.[4]
- **NumPy** : Pour la manipulation et l'analyse des données. NumPy (Numerical Python) est une bibliothèque du langage de programmation Python qui prend en charge les tableaux et matrices multidimensionnels de grande taille, ainsi qu'une vaste collection de fonctions mathématiques de haut niveau permettant d'opérer sur ces tableaux.[25]
- **BeautifulSoup** and **requests** : pour le web scraping.
- **StreamLit** : GUI. StreamLit est une bibliothèque Python permettant de créer des applications web rapides et interactives. [34]
- **MMDetection** : MMDetection est une boîte à outils open-source basée sur PyTorch, conçue pour les tâches de détection d'objets. Elle fournit cadre flexible pour le développement de modèles personnalisés. [46]
- **MMCV** : MMcv est une bibliothèque open-source pour les tâches de vision par ordinateur. MMcv est conçue pour être utilisée avec des frameworks d'apprentissage profond comme PyTorch et TensorFlow[45]

— **Pillow (PIL)** : Python Imaging Library (PIL) permet d'ouvrir, de manipuler et d'enregistrer divers formats de fichiers d'images. [48]
— **OpenVINO** : OpenVINO (Open Visual Inference and Neural Network Optimization) est une boîte à outils complète développée par Intel qui facilite le développement et le déploiement de modèles de vision par ordinateur et d'apprentissage profond. [26]
— **Visual Studio Code (VS Code)** : IDE. VS Code est un éditeur de code développé par Microsoft. Il est conçu pour être utilisé avec un large éventail de langages de programmation, y compris Python. [43]

4.4 Hyperparamètres utilisés

Dans notre architecture initiale, nous avons utilisé un ensemble d'hyperparamètres pour entraîner notre modèle de détection des météores. Les hyperparamètres sont cruciaux car ils influencent le processus d'apprentissage et les performances du modèle. Nous définissons ci-dessous chaque hyperparamètre en détail et présentons un tableau avec les valeurs par défaut utilisées dans nos expériences.

1. **batch_size** (taille_du_lot) :
   — Le nombre d'échantillons d'entraînement utilisés dans une itération. Une taille de lot plus grande peut accélérer l'entraînement et rendre les estimations du gradient plus stables, mais elle nécessite plus de mémoire.
2. **num_workers** (nombre_de_travailleurs) :
   — Le nombre de sous-processus à utiliser pour le chargement des données.
3. **max_epochs** (nombre_max_d'epochs) :
   — Le nombre maximal d'epochs pour l'entraînement. Un epoch est un passage complet à travers l'ensemble de données d'entraînement.
4. **start_factor (LinearLR)** (facteur_de_départ) :
   — Le facteur initial par lequel le taux d'apprentissage est multiplié au début de l'entraînement.
5. **milestones (MultiStepLR)** (jalons) :
   — Epoch spécifiques auxquelles le taux d'apprentissage est ajusté lors de l'utilisation du planificateur MultiStepLR.
6. **gamma (MultiStepLR)** :
   — Le facteur par lequel le taux d'apprentissage est multiplié à chaque Milestone.
7. **lr (learning rate)** (taux_d'apprentissage) :
   — La taille du pas à chaque itération lors de la minimisation de la fonction de perte.
8. **momentum (SGD)** (momentum) :
   — Un paramètre qui aide à accélérer les vecteurs de gradient dans la bonne direction, conduisant ainsi à une convergence plus rapide.
9. **weight_decay (SGD)** (décroissance_du_poids) :
   — Un paramètre de régularisation qui prévient le surapprentissage en ajoutant une pénalité sur les poids importants.
10. **loss_weight** (poids_de_la_perte) :

— Le poids attribué à une fonction de perte particulière. Il détermine l'importance relative des différents termes de perte lors de l'optimisation du modèle.

11. **max_per_img (rcnn/rpn)** (max_par_image) :
    — Le nombre maximal de propositions ou de détections à considérer par image pendant l'entraînement ou l'inférence. Il limite le nombre de boîtes englobantes traitées, réduisant la complexité de calcul.
12. **shuffle (sampler)** :
    — Un booléen indiquant si le jeu de données doit être mélangé pendant l'entraînement. Le mélange aide à briser tout ordre inhérent dans les données, conduisant à une meilleure généralisation du modèle.

Le Tableau 4.1 présente les valeurs des hyperparamètres par défaut utilisés.

Hyperparamètere	**Valeur par défaut**
batch_size (train)	2
batch_size (validation)	1
num_workers	2
max_epochs	12
start_factor (LinearLR)	0.001
milestones (MultiStepLR)	[8, 11]
gamma (MultiStepLR)	0.1
lr (SGD)	0.02
momentum (SGD)	0.9
weight_decay (SGD)	0.0001
type (optimizer)	'SGD'
type (sampler)	'DefaultSampler'
loss_weight (loss functions)	1.0
max_per_img (rcnn)	100
max_per_img (rpn)	1000
shuffle (sampler)	True

TABLE 4.1 – Default Hyperparameter Values

4.5 Mesures d'évaluation

4.5.1 COCO Metrics (Métriques COCO)

1. **Average Precision (AP)** (Précision Moyenne)
    — **Définition** : La Précision Moyenne est la métrique principale utilisée en détection d'objets pour mesurer la précision et le rappel du modèle. Elle est calculée comme l'aire sous la courbe précision-rappel.
    — **Calcul** : L'AP est calculée pour différents seuils d'Intersection sur Union (IoU), qui mesurent le chevauchement entre les boîtes englobantes prédites et les boîtes de vérité terrain.

- **AP@[IoU=0.50 :0.95]** : C'est la moyenne de l'AP sur des seuils d'IoU allant de 0.50 à 0.95 par incréments de 0.05. Elle fournit une mesure complète des performances du modèle.
- **AP@[IoU=0.50]** : C'est l'AP à un seuil d'IoU de 0.50, qui est une mesure plus indulgente de la précision.
- **AP@[IoU=0.75]** : C'est l'AP à un seuil d'IoU de 0.75, qui est une mesure plus stricte de la précision.

2. **Average Recall (AR)** (Rappel Moyen)
   - **Définition** : L'AR mesure le rappel du modèle à différents seuils d'IoU.
   - **Calcul** : L'AR est calculé en faisant la moyenne du rappel à différents seuils d'IoU.
   - **AR@[maxDets=100]** : Cela mesure le rappel en considérant les 100 meilleures détections par image.

3. **Intersection over Union (IoU)** (Intersection sur Union)
   - **Définition** : L'IoU est une métrique utilisée pour évaluer le chevauchement (intersection) entre les boîtes englobantes prédites et les boîtes de vérité terrain. La Figure 4.1 explique comment calculer l'IoU.

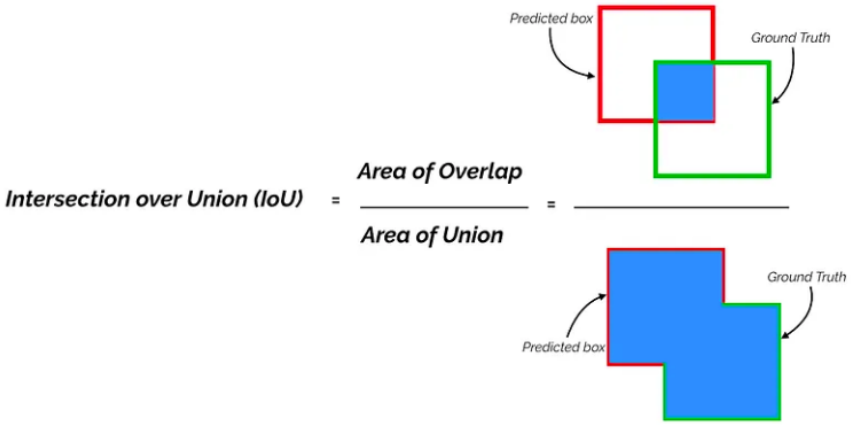

FIGURE 4.1 – Exemple expliquant la manière dont l'Intersection sur Union est calculée [6]

- **Usage** : Différents seuils d'IoU (par exemple, 0.50, 0.75) sont utilisés pour évaluer la précision et le rappel du modèle.

La Figure 4.2 montre l'importance du choix du seuil d'IoU sur la détection.

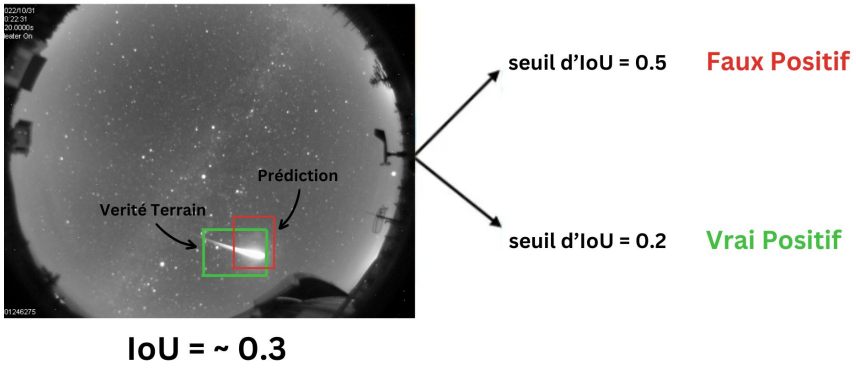

FIGURE 4.2 – Exemple identifiant les vrais positifs et les faux positifs à travers différents seuils en fonction de la tâche

4. **Précision**
   - **Définition** : La Précision est le rapport entre les détections positives vraies et le nombre total de détections positives (vrais positifs + faux positifs).
   - **Calcul** : La Précision est calculée comme suit :

$$\text{Précision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

   où TP = Vrais Positifs (Prédit comme positif et était correct) et FP = Faux Positifs (Prédit comme positif mais était incorrect).

5. **Recall** (Rappel)
   - **Définition** : Le Rappel est le rapport entre les détections positives vraies et le nombre total de positifs réels (vrais positifs + faux négatifs).
   - **Calcul** : Le Rappel est calculé comme suit :

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

   où TP = Vrais Positifs (Prédit comme positif et était correct) et FN = Faux Négatifs (Échec de prédiction d'un objet qui était présent).

4.5.2 Loss Metrics (Métriques de Perte)

1. **Total Loss ('loss')** (Perte Totale) : C'est la perte cumulative pour chaque itération, reflétant les performances globales du modèle. Des valeurs de perte totales plus faibles indiquent de meilleures performances.
2. **RPN Classification Loss ('loss_rpn_cls')** (Perte de Classification RPN) : Cela mesure la perte associée à la tâche de classification dans le Réseau de Propositions de Régions (RPN). Cela aide à comprendre la qualité de l'identification des objets par le RPN.

3. **RPN Bounding Box Loss (‘loss_rpn_bbox‘)** (Perte de boîte englobante RPN) : Cela indique la perte pour la tâche de régression des boîtes englobantes dans le RPN. Cela mesure la précision avec laquelle le RPN prédit les coordonnées des boîtes englobantes.
4. **Classification Loss (‘loss_cls‘)** (Perte de Classification) : Cela mesure la perte liée à la tâche de classification finale, où le modèle classe les objets dans des catégories prédéfinies.
5. **Bounding Box Loss (‘loss_bbox‘)** (Perte de boîte englobante) : Cela indique la perte pour la tâche de régression des boîtes englobantes dans la tête de détection finale. Cela mesure la précision avec laquelle le modèle prédit les coordonnées des boîtes englobantes pour les objets détectés.

4.5.3 Accuracy (Précision)

— **Accuracy (‘acc‘)** (Précision) : Cette métrique indique la précision de classification du modèle à chaque itération. C'est le pourcentage d'objets correctement classifiés sur le nombre total d'objets. Une précision plus élevée indique de meilleures performances.
— **Calcul :** Elle est calculée de la manière suivante :

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

où TP = Vrais Positifs, FP = Faux Positifs, TN = Vrais Negatifs et FN = Faux Négatifs.

4.6 Résultats des Phase d'entraînement avec différents modèles

4.6.1 Architecture initiale : Faster RCNN avec ResNet Backbone

Nous avons initié nos expériences avec l'architecture Faster RCNN utilisant un backbone ResNet. Ce modèle a servi de référence pour les améliorations ultérieures.

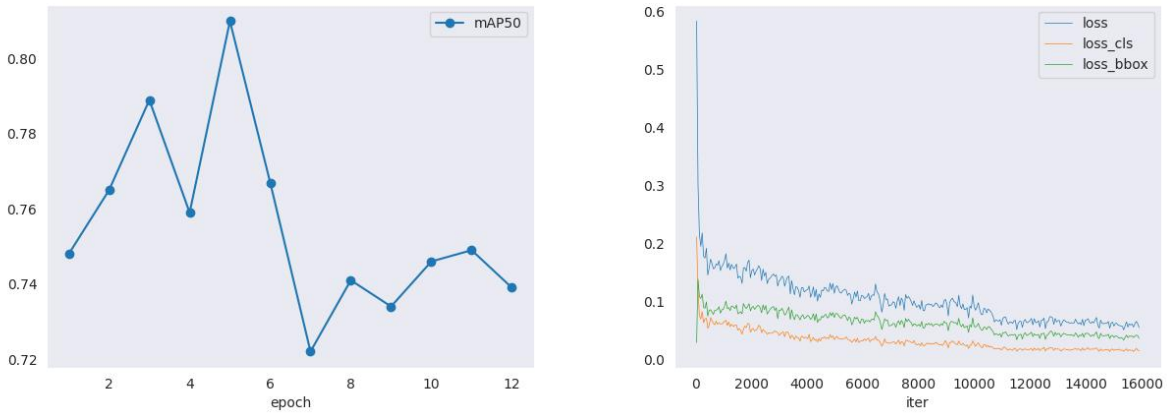

FIGURE 4.3 – Résultats mAP de Faster RCNN avec backbone ResNet (gauche) et Perte de Faster RCNN avec backbone ResNet (droite).

Le graphique de la Figure 4.3 (à gauche) montre la progression de la precision durant l'apprentissage de ce modèle. On remarque une fluctuation qui se stablise a la fin de l'entrainement et termine avec un mAP de 0.74 (avec un seuil IoU 0.5).

Pendant ce temps, le graphique montré en Figure 4.3 (à droite) représente la diminution de la perte durant l'entraînement. La perte générale, la perte de classification et la perte de la boîte englobante sont faibles à la fin de l'entraînement (16000 itérations).

FIGURE 4.4 – Résultats de détection sur des images empilées en utilisant Faster RCNN avec backbone ResNet.

La Figure 4.4 représentent les résultats de détection sur des images RGB empilés de météores (qui composent la traînée du météore). Bien que le modèle n'a pas été entraîné pour cette tache, il a réussi détecter certaine traînée de météores sur des images en couleurs.

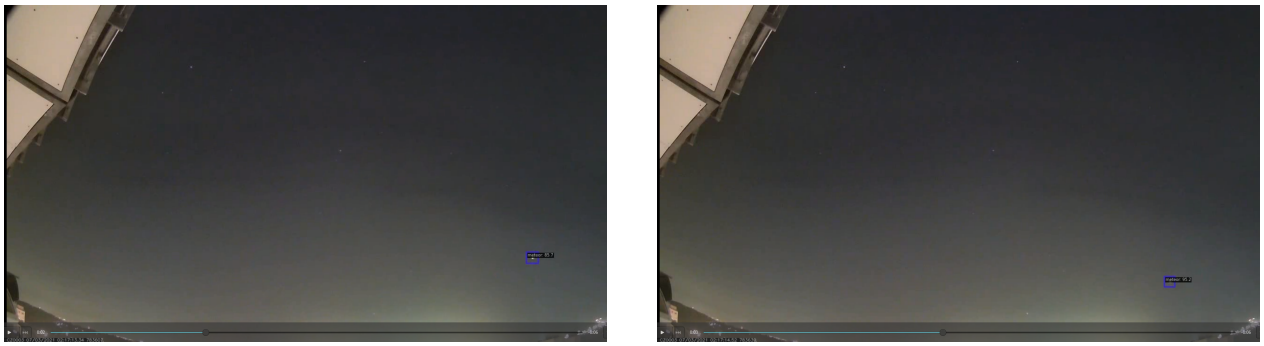

FIGURE 4.5 – Résultats de détection de la vidéo 1 (sur différents frames) utilisant Faster RCNN avec backbone ResNet.

La Figure 4.5 présente des résultats de détection sur une vidéo RGB hors de notre dataset utilisée pour tester la performance du modèle. On peut observe que le modèle est capable de généraliser sur des données qui ne ressemble pas a ceux de l'apprentissage.

FIGURE 4.6 – Résultats de détection de la vidéo 2 (sur 3 différents frames) utilisant Faster RCNN avec backbone ResNet.

Une autre vidéo (Video 2) pour tester la performance du modèle. Dans ces frames 4.6 on peut observer que le modèle a pu identifier la lune et les lumières de la ville comme des non-météores tout en identifiant le météore lorsqu'il est passé dans le ciel avec une confiance de 70%.

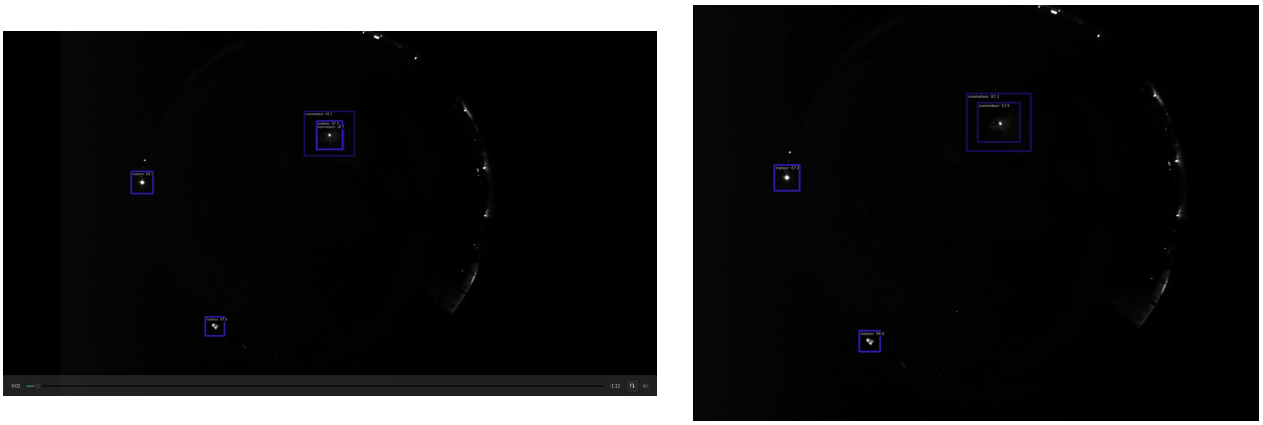

FIGURE 4.7 – Résultats de détection de la vidéo 3 utilisant (differents frames) Faster RCNN avec backbone ResNet.

La Figure 4.7 présente les résultats de détection de notre modèle en utilisant une autre vidéo (vidéo 3) elle a été spécialement choisie pour tester le modèle car elle est la plus difficile. Cette vidéo contient de nombreux objets non-météores et un météore très petit et de faible intensité lumineuse passant au bas de la vidéo. L'architecture n'a pas pu détecter le météore dans cette vidéo et a classé certains non-météores comme météores.

Le modèle proposé démontre de bonnes performances en détection de météores mais a du mal avec les images empilées montrant les traînées complètes des météores. Par conséquent, nous avons opté pour tester un autre backbone.

4.6.2 Transition vers Faster RCNN avec Swin Transformer Backbone

Reconnaissant le potentiel des modèles basés sur les transformateurs, nous sommes passés à Faster RCNN avec un backbone Swin transformer. Cette version a montré une amélioration notable de la précision de détection et de la robustesse.Cette version a été utilisée avec le premier dataset contenant seulement 1678 images entre météores et non-météores sans augmentation de données.

La Figure 4.8 (à gauche) montre les résultats de mAP (@ 0.5) pendant les 12 epochs d'entraînement du Faster RCNN avec le backbone Swin. Le mAP a diminué par rapport à la dernière version avec le backbone ResNet. Nous observons aussi sur la Figure 4.8 (à droite) une diminution de la perte générale, de la perte de la boîte englobante et de la perte de classification pendant les itérations d'entraînement.

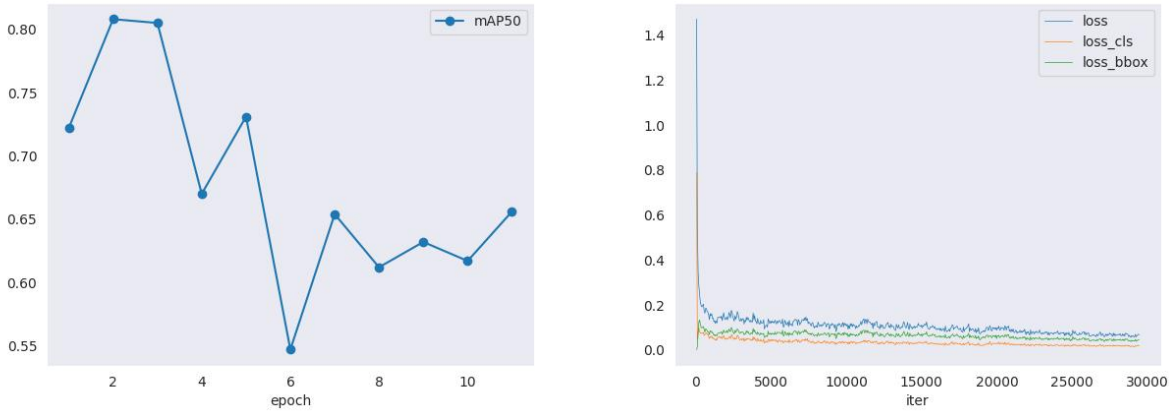

FIGURE 4.8 – Résultats mAP + Perte du Faster RCNN avec backbone Swin.

FIGURE 4.9 – Résultats de détection de la vidéo 1 (différents frames) utilisant Faster RCNN avec backbone Swin.

La Figure 4.9 montrent la détection du météore dans différents frames la vidéo 1 avec une confiance allant de 93 à 99,6 pendant le passage de ce météore.

FIGURE 4.10 – Résultats de détection de la vidéo 2 (différents frames) utilisant Faster RCNN avec backbone Swin.

On observe sur la Figure 4.10 le résultats de detection sur la video 2. La confiance du météore détecté variait entre les frames de 75 à 99, et les non-météores (lune et lumières de la ville) entre 80 et 99 de confiance.

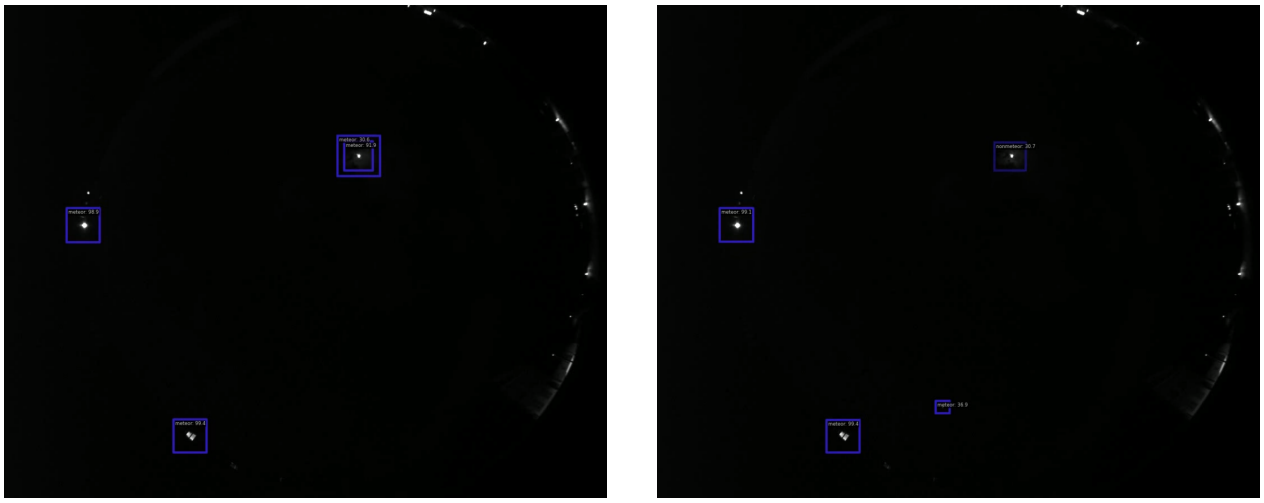

FIGURE 4.11 – Résultats de détection de la vidéo 3 (différents frames) utilisant Faster RCNN avec backbone Swin.

La Figure 4.11 montre les résultats de détections sur quelques frames de la vidéo 3. Le modèle a pu détecter le petit météore qui passe au milieu en bas, mais seulement sur un seul frame, tandis qu'il a détecté certains non-météores comme des météores et nous pouvons voir des redondances dans les boîtes englobantes. Ceci est bien meilleur qu'avec le modèle précédant de Faster RCNN + ResNet backbone.

4.6.2.1 Modification Itérative et Résultats

1. **Augmentation des Données de Validation et de Test :**

— L'objectif est d'atteindre une proportion équilibrée entre les sources de données de test et de validation.
— Nous pouvons observer l'amélioration du mAP @ 0.5 sur la Figure 4.12 par rapport à la dernière variation du Faster RCNN avec Swin Backbone, et c'est grâce aux techniques d'augmentation de données que nous avons choisies et à la proportion de sources de données entre les données de validation et d'entraînement, à la fin elle atteint un mAP = 0.75

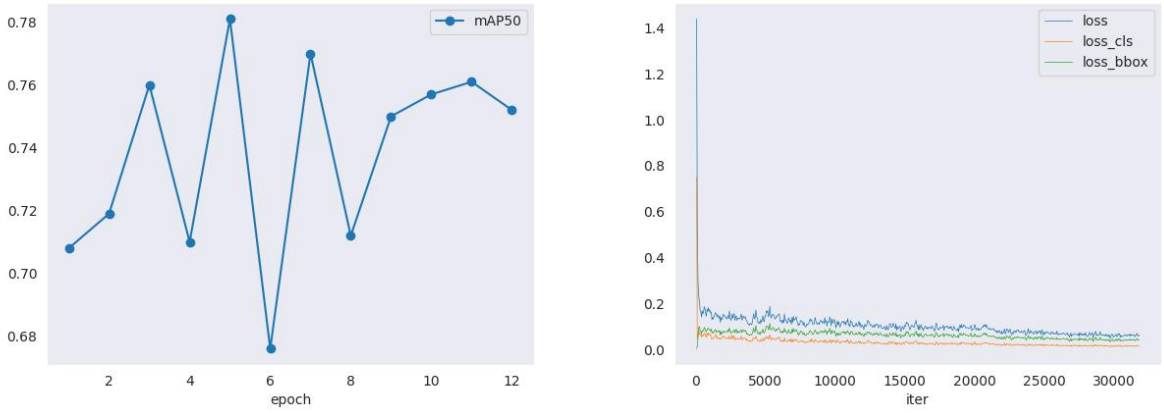

FIGURE 4.12 – Résultats mAP + Loss de Faster RCNN avec Swin Backbone sur les Données Augmentées.

— Le test de cette nouvelle configuration du modèle donne de bons résultats sur les images empilées de météores, et fait plus de détection que la version antérieur. Ces résultats sont illustrés dans la Figure 4.13.

FIGURE 4.13 – Résultats de Détection de l'Image Empilée en utilisant Faster RCNN avec Swin Backbone sur les Données Augmentées.

FIGURE 4.14 – Résultats de Détection de la Vidéo 1 en utilisant Faster RCNN avec Swin Backbone Entraîné sur les Données Augmentées.

— On observe sur la Figure 4.14 les résultats de détection de la vidéo 1. On remarque de fausses détections. Comme nous pouvons le voir avec le point noir dans le coin supérieur gauche des deux images de la figure, a été détectée comme un météore avec une confiance de 48 tandis que le météore réel a été détecté avec une confiance de 99,7 sur l'image de droite.

2. **Augmentation des epochs et de la Taille du Batch :**
   — Pour améliorer le processus d'apprentissage du modèle, nous avons expérimenté en augmentant la taille du batch et en prolongeant la durée de l'entraînement à 25 epochs. Cela nécessitait un matériel plus puissant pour gérer la charge computationnelle accrue.

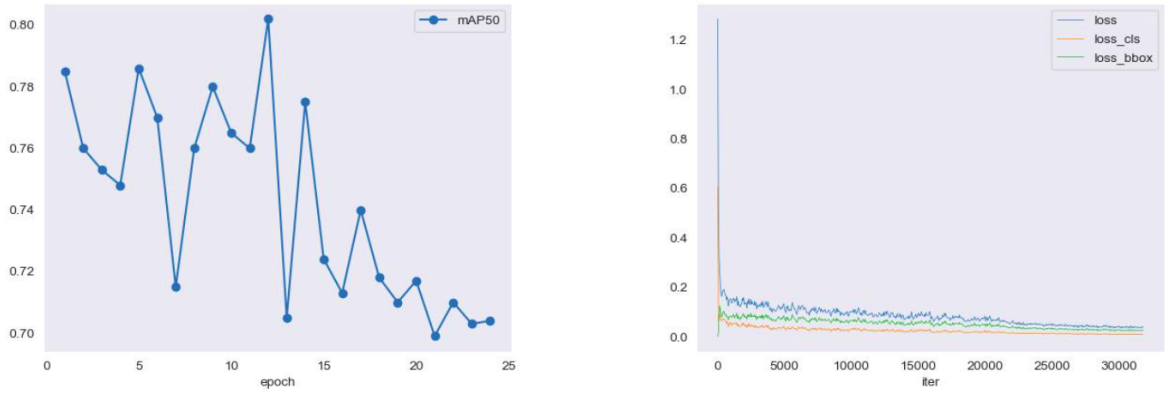

FIGURE 4.15 – Faster RCNN avec Swin Backbone + Modification d'epoch & Taille du Batch mAP Résultats.

— La Figure 4.15 illustre les résultats de l'entraînement de Swin avec plus d'epoch. Nous observons une légère dégradation des performances. Les valeurs mAP se sont stabilisées entre 0,72 et 0,74 à partir des epochs 15 à 25, suggérant que le modèle convergeait plus tôt. Sur la base de ces observations, nous sommes revenus à 12

epochs, car un entraînement supplémentaire n'améliorait pas les performances et ajoutait seulement une charge computationnelle supplémentaire.

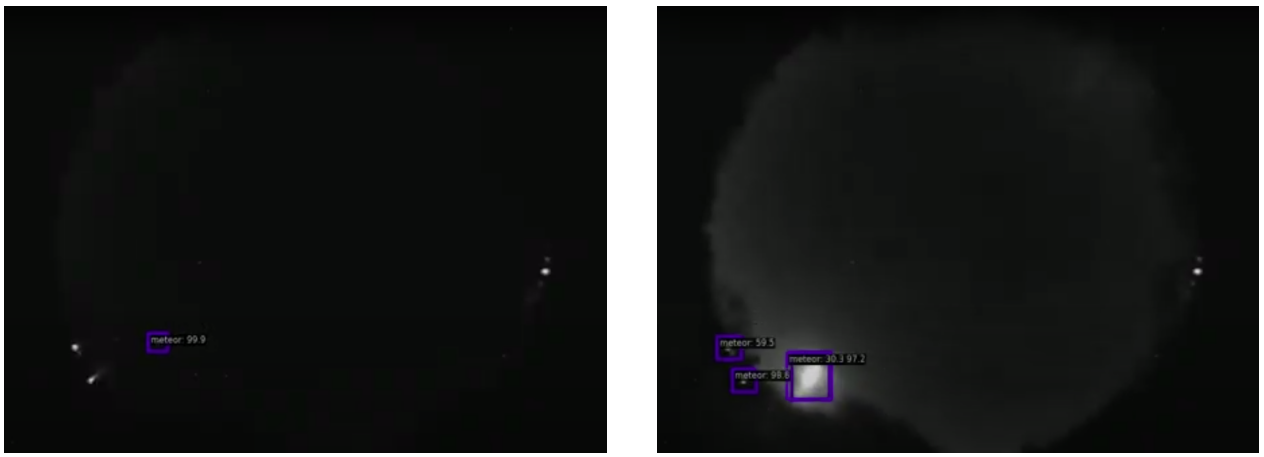

FIGURE 4.16 – Résultats de Détection de la Vidéo 4 (differents frames) en utilisant Faster RCNN avec Swin Backbone + Modification d'epoch & Taille du Batch.

— La Figure 4.16 illustre les résultat de détection sur deux differents frames de la vidéo 4 testée pour cette variation du modèle. On observe a gauche que le météore avec une haute confiance, suivi plus tard de la détection de faux positifs de non-météores en tant que météores (a droite).

— À ce stade, notre objectif est clair : enrichir l'ensemble de données avec des sources diverses pour améliorer la capacité du modèle à généraliser. Nous avons décidé de revenir à 12 epochs et à la taille de batch initiale de 2. Nos tentatives précédentes avec des epochs étendues et des tailles de batch plus grandes ont montré que les performances du modèle avaient stagné. En incorporant des données de différentes sources, nous avons cherché à améliorer la polyvalence du modèle.

3. **Incorporation des données canadiennes et augmentation avancée des données :**
   — Pour amener les performances de notre modèle à un niveau supérieur, nous nous sommes tournés vers l'intégration des données canadiennes. Ces données apportaient une perspective nouvelle, différente des autres sources, ce qui s'est avéré précieux pour améliorer les capacités de généralisation du modèle. Parallèlement, nous avons utilisé des techniques d'augmentation (flip, rotation) pour diversifier davantage les entrées d'entraînement.
   — Avec l'ensemble de données V2, enrichi des images distinctives du Canada, la capacité du modèle à généraliser s'est nettement améliorée. Cette variation des données était cruciale, fournissant au modèle une gamme plus large de scénarios à apprendre.

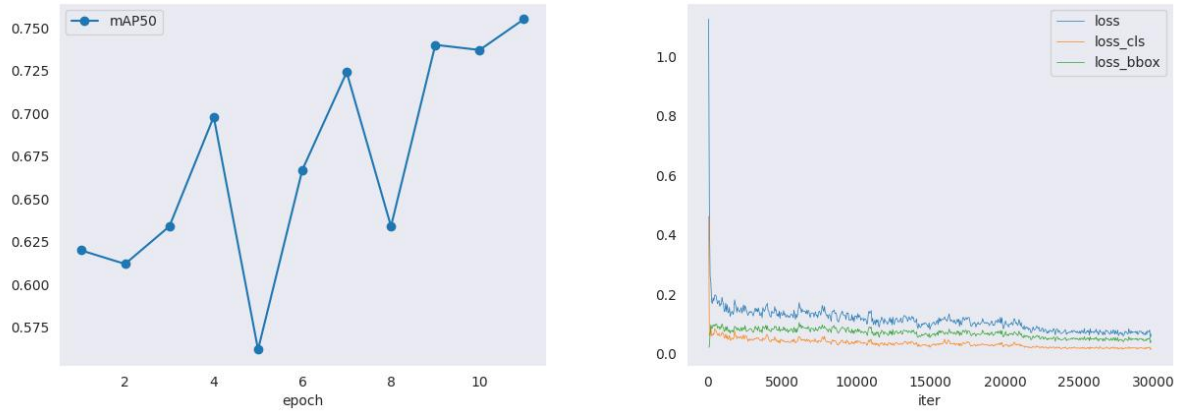

FIGURE 4.17 – Faster RCNN avec Swin Backbone sur les résultats mAP et Perte de l'ensemble de données V2.

— On remarque sur la Figure 4.17 que la mAP a augmenté et que la courbe est bien meilleur que celle du modèle précédant. soulignant la performance améliorée et la généralisation du modèle.

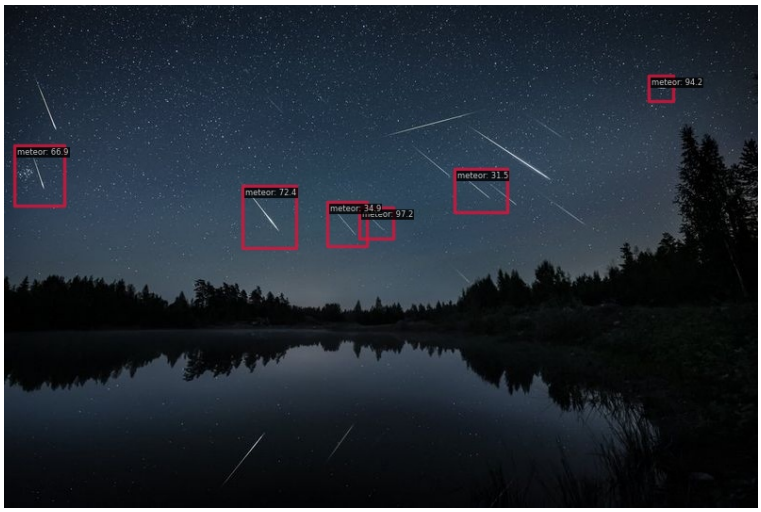

FIGURE 4.18 – Résultats de détection de l'image empilée en utilisant Faster Rcnn avec Swin Backbone sur l'ensemble de données V2.

— La Figure 4.18 montre le résultat de détection de cette dernière configuration de swin, et on remarque une nette amélioration et une meilleure détection.

4. **Optimisation de la détection des boîtes englobantes (NMS et seuils IoU) :**
— Notre modèle montre déjà des résultats prometteurs, nous concentrons maintenant nos efforts sur l'optimisation de la détection des boîtes englobantes pour améliorer encore ses performances. Cette prochaine phase consiste à peaufiner la Suppression Non Maximale (NMS) et les seuils d'Intersection sur Union (IoU), des ajustements cruciaux pour détecter avec précision de petits objets tels que les météores.

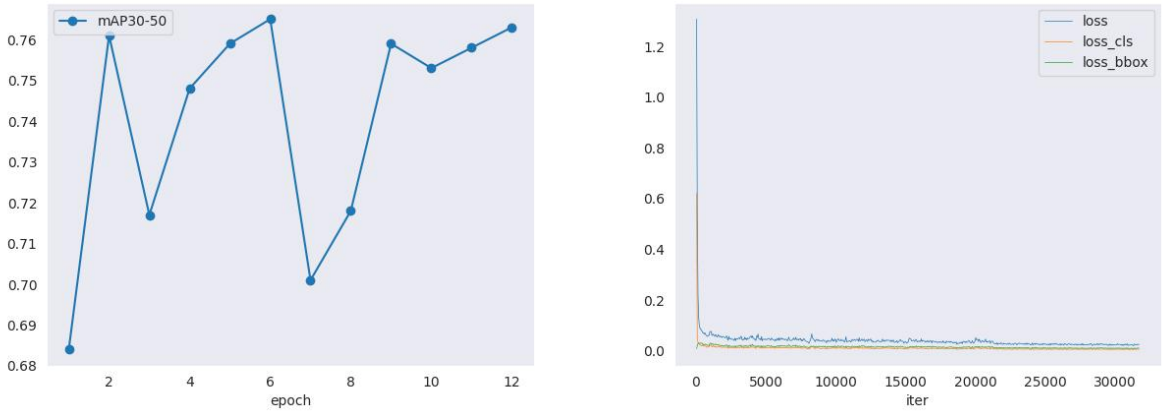

FIGURE 4.19 – MAP et Perte de Faster RCNN avec Swin Backbone + Ajustement NMS & IoU.

— Reconnaissant les défis uniques posés par ces petits objets, nous avons ajusté les paramètres NMS et IoU. Notre objectif est de favoriser la Précision Moyenne de Précision (mAP) dans la plage de 30 à 50, comme le montre la Figure 4.19.

— En ajustant le seuil IoU de la NMS à 0,45 pour la tête de Région d'Intérêt (ROI) et à 0,6 pour le Réseau de Proposition de Région (RPN), nous avons réduit efficacement les boîtes englobantes redondantes et amélioré la précision de nos détections. Ces modifications, spécifiquement adaptées à la détection d'objets de petite taille, ont non seulement renforcé la précision du modèle, mais ont également considérablement amélioré sa robustesse.

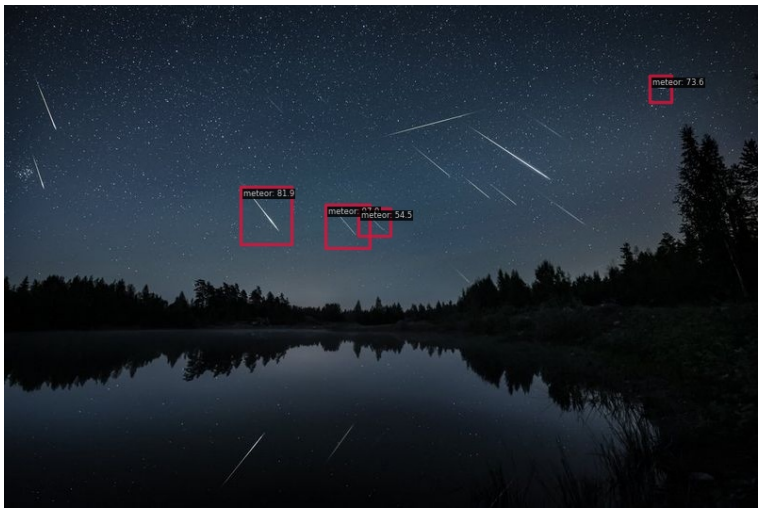

FIGURE 4.20 – Résultats de détection de l'image empilée en utilisant Faster Rcnn avec Swin Backbone + Ajustement NMS & IoU.

— La Figure 4.20 montre les résultats de détection sur un image empilée. On remarque que les modifications apportées nous permettent de détecter plusieurs météores.

— Les données de nos expériences ont confirmé ces améliorations. Les métriques de perte de notre modèle ont montré une réduction significative : la perte totale est passée à environ 0,0549 comme l'illustre la Figure 4.19.
— La précision a augmenté à un impressionnant 99,6%, et les scores mAP ont démontré la capacité améliorée du modèle : un mAP30-50 de 0,763 mAP50 de 0,7.

5. **Ajustement du Taux d'Apprentissage et du Weight decay :**
— Dans notre quête pour optimiser notre modèle de détection de météores, nous nous tournons vers le fine-tuning du taux d'apprentissage et du Weight decay. Cette étape peut faire une différence substantielle dans les performances et la généralisation d'un modèle. Explorons pourquoi nous avons choisi ces modifications spécifiques et comment nos choix architecturaux ont façonné cette décision.
Le Tableau 4.2 montre les valeurs des paramètres modifiés :

Paramètre	**Valeur**
Taux d'apprentissage (Learning Rate)	0.0001
Poids de la décroissance (Weight Decay)	0.0001

TABLE 4.2 – Valeurs du Taux d'apprentissage et du Weight decay dans cette Variation du Modèle

— **Pourquoi Ces Modifications ?**
— Imaginez-vous naviguer sur un fil tendu – le taux d'apprentissage est similaire à la vitesse à laquelle vous faites chaque pas. Avancez trop vite, et vous risquez de tomber ; avancez trop lentement, et vous pourriez ne jamais atteindre la fin. En fixant le taux d'apprentissage à 0.0001, notre modèle a pu apprendre efficacement sans dépasser les poids optimaux, garantissant une progression équilibrée et constante.

— Weight Decay agit comme une main directrice, vous empêchant de dévier trop loin du cours. En pénalisant les grands poids, elle maintient la complexité du modèle sous contrôle, améliorant sa capacité à généraliser des données d'entraînement à des scénarios non vus. Une Weight Decay de 0.0001 a trouvé le juste équilibre, évitant le surapprentissage et assurant des performances robustes sur des ensembles de données diversifiés.

— Le Transformer Swin, a joué un rôle crucial dans le succès du modèle. Explorons les composants de sa conception qui ont été modifiées à cette partie sur Le Tableau 4.3.

Composant	**Configuration**
Taux de Drop Path	0.2
Optimiseur	AdamW ; Taux d'apprentissage : 0.0001 ; Poids de la décroissance : 0.0001 ; Betas : (0.9, 0.999)
Scheduler	LinearLR : Facteur de départ 0.001, fin à 1000 itérations ; MultiStepLR : Jalons aux epochs 8 et 11, gamma 0.1

TABLE 4.3 – Nouvelle Configuration du Transformer Swin

— **Résultats :**

Le Tableau 4.4 et la Figure 4.21 montre les résultats de la nouvelle configuration.

Métrique	**Valeur**
Perte	0.056008519642055034
Perte (RPN Cls)	2.597205937149738e-05
Perte (RPN Bbox)	0.003291938677430153
Perte (Cls)	0.01633708234410733
Précision	99.51171875 %
Perte (Bbox)	0.036353526264429094
mAP @ 30-50	0.792
mAP (moyen)	0.84
mAP (grand)	0.988

TABLE 4.4 – Résultats de la configuration après 12 epochs d'Entraînement

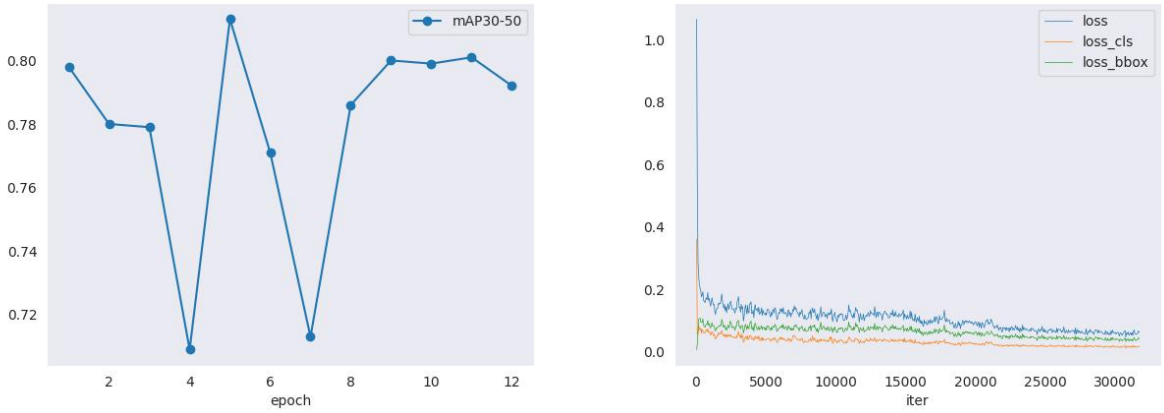

FIGURE 4.21 – mAP et perte Faster RCNN avec Swin Backbone + Ajustement du Taux d'apprentissage & du Weight decay.

FIGURE 4.22 – Résultats de détection de la Vidéo 1 en utilisant Faster Rcnn avec Swin Backbone + Ajustement du Taux d'apprentissage & du Weight decay.

— Les résultats du Tableau 4.4 et la Figure 4.21 ont mis en évidence la performance robuste du modèle. La précision de 99.51% et un mAP de 0.792 ont démontré l'efficacité de nos ajustements d'hyperparamètres. Les scores mAP pour différentes tailles d'objets (petits, moyens et grands) ont également indiqué la polyvalence et la précision du modèle dans la détection de météores de différentes échelles, comme le montrent les résultats de détection des Vidéos 1 & 2 - Figure 4.22 -, où le modèle a également identifié avec succès tous les non-météores.

6. **Ajustement de l'Ordonnancement Dynamique et du Dropout :**

Dans notre effort continu pour améliorer les performances de notre modèle de détection de météores, nous avons expérimenté avec un ordonnancement dynamique et un dropout supplémentaire. L'hypothèse est que ces ajustements pourraient potentiellement renforcer la capacité du modèle à généraliser et à prévenir le

surapprentissage. Cependant, nos résultats suggèrent le contraire, nous conduisant à revenir à la configuration précédente. Le Tableau 4.5 montre les modifications sur les valeurs de Drop Path et Drop de l'Attention du backbone Swin.

Paramètre	**Config Précédente**	**Config avec Dropout**
Taux de Drop de l'Attention	0.0	0.1
Taux de Drop Path	0.2	0.3

TABLE 4.5 – Paramètres d'Ajustement de l'Ordonnancement Dynamique et du Dropout

— **Résultats de l'Expérience**

Malgré nos grands espoirs, les ajustements dans l'ordonnancement dynamique et le dropout n'ont pas produit les améliorations significatives que nous anticipions. Les résultats peuvent etre observés sur Le Tableau 4.6 :

— **Métriques de Performance :**

Métrique	**Valeur**
Perte	0.09865216046571731
Perte (RPN Cls)	0.00018997269261149087
Perte (RPN Bbox)	0.005094063326250762
Perte (Cls)	0.028452343940734863
Précision	99.51171875 %
Perte (Bbox)	0.06491578087210655
mAP	0.775
mAP@50	0.743
mAP (moyen)	0.777

TABLE 4.6 – Métriques de Performance avec la Nouvelle Configuration

Bien que les modifications n'aient pas dégradé les performances de manière drastique, on peut observer dans Le Tableau 4.6 les différences marginales indiquent que nos paramètres précédents étaient déjà bien optimisés. Les taux de dropout plus élevés visant à réduire le surapprentissage n'ont pas entraîné de gains significatifs dans des métriques de performance telles que mAP ou la réduction de la perte.

— **Comparaison des modele avec et sans Dropout**

Dans cette partie nous comparons les résultats de deux configuration avec et sans Dropout. Les performances des ces architectures sont représenté sur Le Tableau 4.7.

Les résultats ont clairement démontré que l'introduction de l'ordonnancement dynamique et du dropout supplémentaire n'a pas apporté un avantage significa-

Métrique	Configuration Précédente (sans)	Nouvelle Configuration (avec Dropout)
Perte	0.056	0.098
Précision	99.51%	99.51%
mAP	0.792	0.775

TABLE 4.7 – Comparaison des modèles avec et sans Dropout

tif. La précision est restée largement inchangée, tandis que la perte a légèrement augmenté et mAP a connu une légère baisse.

4.6.3 Analyse et synthèse

4.6.3.1 Comparaison des Performances du Modèle

Nous présentons une comparaison détaillée des différentes versions du modèle, en mettant en évidence l'impact de chaque modification. Le Tableau 4.8 résument les principales métriques et indicateurs de performance.

Version du Modèle	**mAP@30-50**	**mAP@0.50**	**Échantillons**	**Sources de Données**	**Epoch**	**Learning Rate**
Faster RCNN + ResNet Backbone	-	0.74	1687	Dataset Original sans Augmentation	12 & 16000 iters	0.02
Faster RCNN + Swin Transformer	-	0.65	1687	Dataset Original sans Augmentation	12 & 30000 iters	0.001
Faster RCNN + Swin + Données Validation et Test Augmentées	-	0.75	5321	Dataset V1	12 & 30000 iters	0.001
Faster RCNN + Swin + 20 epochs	-	0.73	5321	Dataset V1	25 & 30000 iters	0.001
Faster RCNN + Swin + Données Canadiennes et Augment.	-	**0.755**	5651	Dataset V2	12 & 30000 iters	0.001
Faster RCNN + Swin + Ajustement NMS et IOU	0.763	0.7	5651	Dataset V2	12 & 30000 iters	0.001
Faster RCNN + Swin + Ajustement LR et WD	**0.792**	0.73	5651	Dataset V2	12 & 30000 iters	0.0001
Faster RCNN + Swin + Ordonnancement Dynamique et Dropout	0.775	0.743	5651	Dataset V2	12 & 30000 iters	0.0001

TABLE 4.8 – Comparaison des Différentes Versions du Modèle et Leurs Métriques de Performance

4.6.4 Discussion

L'Évolution de la Détection des Météores : Un Parcours d'Innovation et de Précision

Notre exploration pour améliorer la détection des météores a commencé avec une approche fondamentale utilisant le Faster RCNN avec un backbone ResNet. Ce modèle, entraîné sur notre dataset original, a servi de point de départ robuste mais a laissé une grande marge d'amélioration avec un mAP@0.50 de 0.74.

Le Saut vers le Swin Transformer

Passer au Swin Transformer a marqué un changement significatif. Malgré une baisse initiale des performances (mAP@0.50 de 0.65), cette transformation a jeté les bases pour des améliorations futures. L'intégration de données de validation et de test augmentées a élargi notre dataset de 1,687 à 5,321 échantillons, boostant instantanément le mAP@0.50 à 0.75. Ce saut a mis en évidence la puissance de la diversité des données pour entraîner des modèles robustes.

Extension des epochs

Prolonger l'entraînement à 20 epochs a testé la résilience de notre approche. Le modèle a montré une stabilité avec un mAP@0.50 de 0.73, prouvant que l'entraînement supplémentaire pouvait consolider les gains obtenus par l'augmentation.

Augmentation des Données Canadiennes :

L'intégration des données canadiennes a enrichi notre dataset à 5,651 échantillons et a légèrement amélioré le mAP@0.50 à 0.755. Cette étape a démontré l'importance des données géographiques diverses pour améliorer la précision de la détection.

Finetuning : NMS et IOU

Ajuster les paramètres de Non-Maximum Suppression (NMS) et d'Intersection over Union (IOU) a conduit à une amélioration modeste du mAP@30-50 à 0.763 et du mAP@0.50 à 0.7. Ces ajustements ont affiné la précision du modèle, réduisant les faux positifs et augmentant les vrais positifs.

Optimisation des Hyperparamètres : Le Facteur Décisif

Le saut le plus significatif est venu avec des ajustements précis du learning rate et du weight decay. Cette version du modèle a atteint un impressionnant mAP@30-50 de 0.792 et mAP@0.50 de 0.73. Cela a montré le rôle crucial de l'affinage des hyperparamètres pour maximiser les performances du modèle.

Ordonnancement Dynamique et Dropout :

Notre dernière expérience avec l'ordonnancement dynamique et le dropout supplémentaire visait à renforcer la robustesse. Bien qu'elle ait maintenu des vrais positifs élevés et des faux positifs faible, l'amélioration de performance attendue n'a pas été réalisée. Cela nous a conduits à revenir à la meilleure configuration étant la précédente.

4.6.4.1 Défis

Nous avons rencontré plusieurs défis, notamment la gestion des faux positifs dans les données augmentées et des boîtes englobantes redondantes. L'affinage précis des hyperparamètres a nécessité des expérimentations soigneuses pour éviter des rendements décroissants.

4.6.4.2 Meilleur Modèle :

Le modèle le plus performant est celui utilisant le Swin Transformer avec des ajustements du learning rate et du weight decay. Atteignant le mAP@30-50 le plus élevé de 0.792 et mAP@0.50 de 0.73, ce modèle a excellé en précision et en rappel, soulignant l'importance des ajustements stratégiques des paramètres.

4.7 Déploiement

Dans cette sous-section, nous discutons le déploiement de notre modèle de détection de météores, en nous concentrant sur la conversion vers OpenVINO et en comparant les tailles de modèles et les temps d'inférence pour différents formats de précision (FP32, FP16).

Après la conversion du modèle vers OpenVINO, nous avons évaluer les tailles des modèles et les temps d'inférence pour les formats de précision FP32 et FP16. Le format FP16 réduit la taille du modèle et accélère l'inférence. Le Tableau 4.9 présente ces résultats.

Précision	**Taille (Mo)**	**inférence CPU (ms)**	**inférence GPU (ms)**
FP32	180	5400	550
FP16	90	5000	500

TABLE 4.9 – Taille et temps d'inférence en millisecondes (ms) du meilleure modèle avec les précisions FP16 et FP32

Nous remarquons dans Le Tableau 4.9 que le déploiement a réduit la taille du modèle à quelques Mégaoctets réduisant ainsi la demande en mémoire. Nous remarquons également que le déploiement permet d'exécuter notre modèle de deep learning sur des machine dépourvus de GPU en temps tout à fait raisonnable. Nous pouvons observer un résultat de détection avec le modèle déployé sur la Figure 4.23.

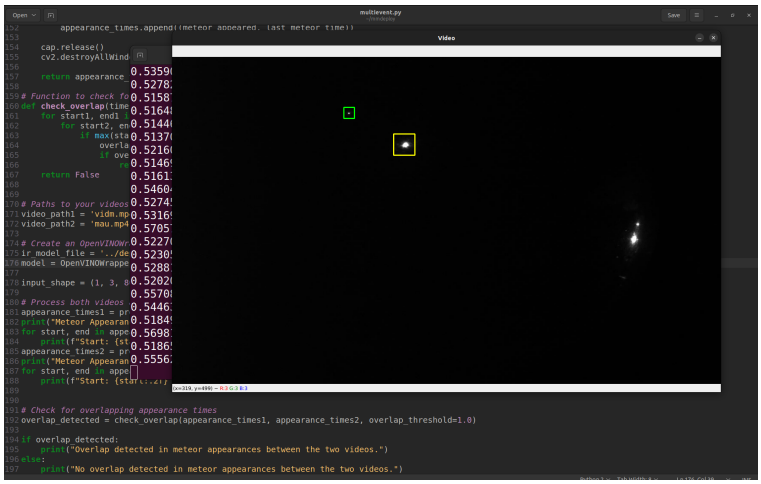

FIGURE 4.23 – Exemple de détection avec le modèle déployé sur une machine avec CPU i5. (Vert représente la classe météore et Jaune la classe non météore.)

4.8 Interface graphique

En plus des résultats expérimentaux étendus et des évaluations de la performance du modèle présentés dans ce chapitre, nous avons également développé une interface graphique conviviale (GUI) pour faciliter l'application pratique de notre système de détection de météores. L'interface graphique a été conçue pour offrir un moyen intuitif et accessible aux utilisateurs d'interagir avec nos modèles et de réaliser la détection de météores sur divers supports.

4.8.1 Fonctionnalités de l'interface graphique

1. **Paramètres ajustables** :
   - **Sélection du modèle** : Les utilisateurs peuvent choisir entre différents modèles de backbone (par exemple, ResNet ou Swin Transformer) pour le processus de détection, leur permettant de tirer avantage du modèle qui correspond le mieux à leurs besoins spécifiques.
   - **Taille de l'entrée** : L'interface graphique permet aux utilisateurs de spécifier les dimensions d'entrée (largeur et hauteur) pour les images ou les vidéos traitées. Cette flexibilité aide à optimiser le processus de détection en fonction de la résolution des données d'entrée.
   - **Seuil de confiance** : Les utilisateurs peuvent ajuster le seuil de confiance, qui détermine le niveau de confiance minimum requis pour qu'une détection soit considérée comme valide. Cela aide à affiner l'équilibre entre rappel et précision.
   - **Classes affichées** : Les utilisateurs peuvent sélectionner les classes à afficher dans les résultats, telles que "Météore" et/ou "Non-Météore". Cette personnalisation permet une détection ciblée en fonction des intérêts de l'utilisateur.

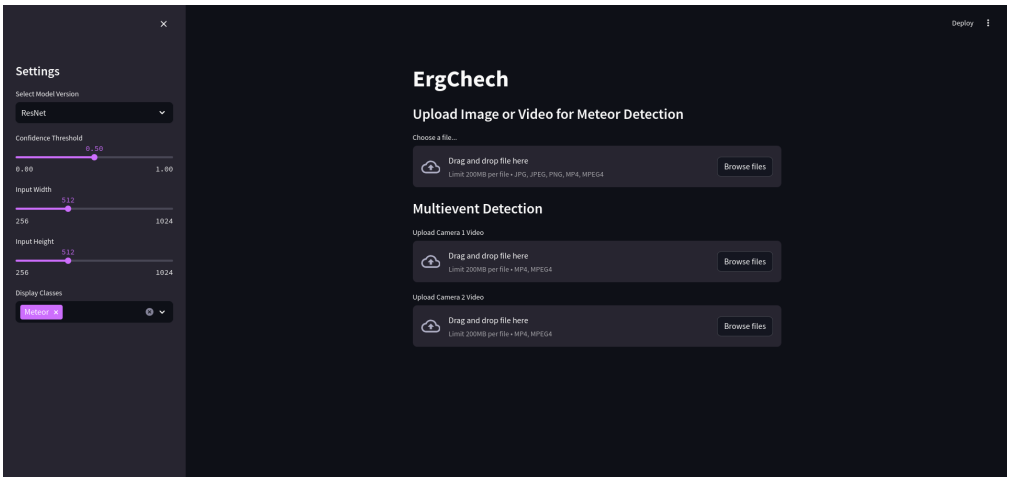

FIGURE 4.24 – Interface graphique principale. L'utilisateur peut ajuster les paramètres dans la barre latérale gauche de l'interface graphique.

2. **Téléchargement et traitement des médias** :
   — **Fonctionnalité de glisser-déposer** : L'interface graphique comprend une section facile à utiliser où les utilisateurs peuvent glisser et déposer des images ou des vidéos pour la détection de météores. Une fois téléchargés, les utilisateurs peuvent cliquer sur un bouton "Commencer le traitement" pour lancer le processus de détection.
   — **Affichage des résultats en temps réel** : Après le traitement, l'interface graphique affiche les médias d'entrée à côté des résultats de la détection, en soulignant les météores et les non-météores détectés avec des boîtes englobantes. Ce retour visuel immédiat aide les utilisateurs à comprendre rapidement les résultats.

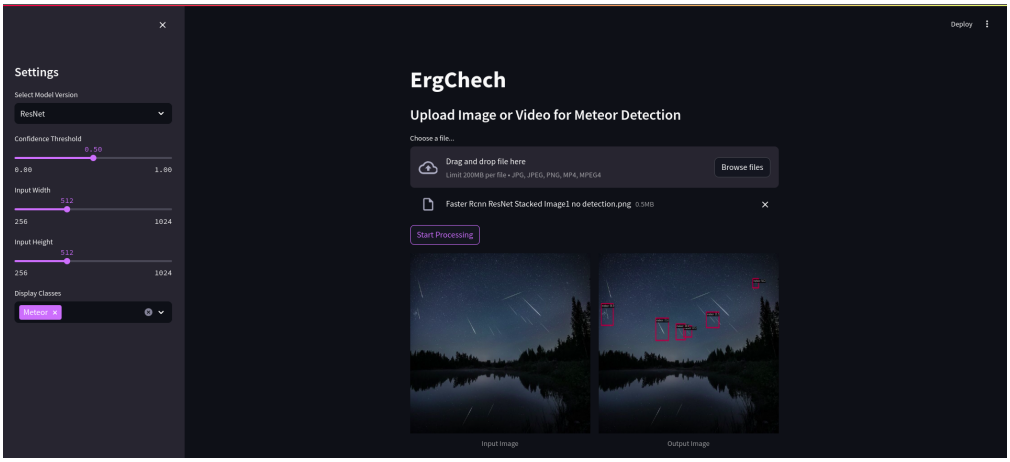

FIGURE 4.25 – Résultats des médias choisis pour être téléchargés et traités pour la détection.

3. **Détection de multi-événements** :
   — **Simulation de l'algorithme de multi-événements** : Étant donné que le réseau algérien de caméras AllSky n'est pas encore entièrement installé, l'interface graphique propose une fonction de simulation pour la détection de multi-événements.

Les utilisateurs peuvent télécharger deux vidéos provenant de différentes caméras AllSky, et le système les analysera pour détecter des événements météoritiques simultanés.

— **Détection synchronisée** : L'interface graphique traite les deux vidéos téléchargées pour identifier et mettre en évidence les météores détectés simultanément dans les deux flux. Cette fonctionnalité est cruciale pour étudier les trajectoires des météores et valider les occurrences de multi-événements.

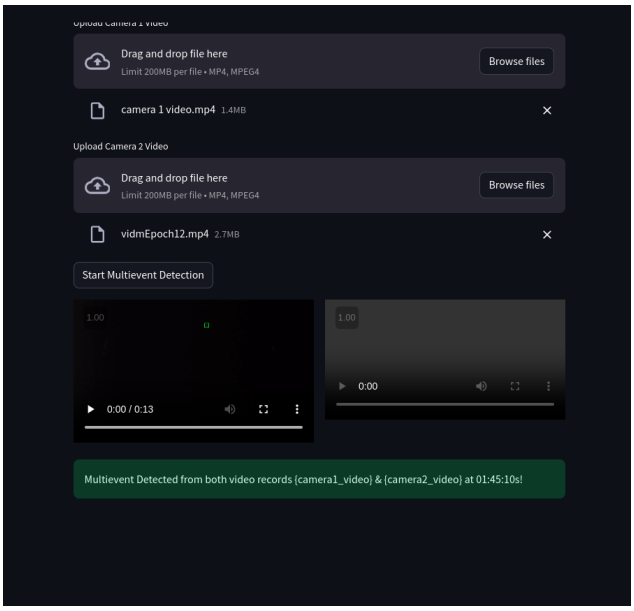

FIGURE 4.26 – Simulation de la détection MultiEvent à l'aide de deux vidéos provenant de caméras différentes pour démontrer le même événement dans les deux flux.

4.9 Conclusion

Dans ce chapitre, nous avons mené de nombreuses expérimentations sur l'architecture Faster RCNN, en utilisant deux variantes principales : ResNet et Swin Transformer comme backbones. Cependant, notre exploration ne s'est pas limitée à ces architectures. Nous avons également expérimenté avec divers hyperparamètres pour optimiser les performances de notre modèle. Les résultats obtenus montrent que l'architecture Faster RCNN avec Swin Transformer surpasse celle avec ResNet en termes de précision et de robustesse, surtout dans des scénarios complexes de détection de météores. Les ajustements minutieux des hyperparamètres, tels que le taux d'apprentissage, la régularisation par décrochage, et les seuils de suppression des non-maximas (NMS), ont permis d'affiner notre modèle pour obtenir les meilleurs résultats possibles. De plus, nous avons quantifié nos modèles en utilisant le format FP16, ce qui a permis de réduire les temps d'inférence tout en maintenant des performances élevées. Cette quantification est particulièrement bénéfique pour des déploiements en temps réel sur des systèmes embarqués. Enfin, pour évaluer les capacités de nos modèles dans des conditions réalistes, nous avons implémenté une simulation de détection multi-événements, démontrant ainsi l'efficacité et la fiabilité de notre approche dans des scénarios pratiques.

Conclusion générale

La détection des météores présente un mélange unique de défis et d'opportunités dans le domaine de l'observation astronomique. Ce mémoire capture l'essence de nos efforts, de la conception initiale à la mise en œuvre finale.

Notre exploration a commencé par une compréhension approfondie du phénomène complexe des météores et des défis inhérents à leur détection. Nous nous sommes penchés sur les méthodologies de l'état de l'art en identifiant leurs limites. Dans ce mémoire nous avons proposé une approche basée sur le deep learning pour la détection de météores.

Nous avons commencé par créer un ensemble de données avec des images provenant de différentes sources. Cet ensemble de données a joué un rôle essentiel dans l'entraînement de nos modèles afin d'obtenir une précision et une fiabilité élevées. Notre choix d'architectures de modèles, y compris Faster RCNN avec ResNet et Swin Transformer, a été motivé par leur efficacité prouvée dans les tâches de détection d'objets.

Grâce à une expérimentation rigoureuse, nous avons affiné nos modèles, en employant des techniques telles que la suppression non maximale (NMS) et l'ordonnancement dynamique. Chaque étape nous a rapprochés de notre objectif, avec des améliorations significatives de la précision et une réduction des faux positifs. Les résultats ont été méticuleusement documentés, fournissant une analyse comparative claire des différentes versions du modèle et de leurs performances.

Nous avons terminé par le développement d'une interface graphique interactive, conçue pour faciliter la tache de détection des météores pour les astrophysiciens. De plus nous avons développer une fonction de détection multi-site, qui synchronise les temps de détections de différentes caméras pour identifier les événements météoriques simultanés.

Notre approche est une contribution pour la détection de météores. Notre système se contente pas seulement de remédier aux limites existantes, il établit également une nouvelle référence pour la recherche et le développement futurs du deep learning dans ce domaine.

Pour l'avenir, les avancées réalisées dans le cadre de ce projet ouvrent la voie à d'autres explorations et améliorations comme le tracking, et la triangulation des coordonnées a l'aide de plusieurs caméras.

Résumé

La détection des météores présente un mélange unique de défis et d'opportunités dans le domaine de l'observation astronomique. Ce mémoire s'engage dans un voyage innovant pour améliorer la détection des météores à l'aide de techniques avancées d'apprentissage profond. Nous nous sommes plongés dans les complexités des météoroïdes, des météores et des météorites, en explorant leur signification scientifique et les complexités liées à leur détection.

Dans notre projet, nous avons constitué un solide ensemble de données et expérimenté des modèles de détection d'objets, notamment Faster RCNN avec ResNet et Swin Transformer backbones. Notre approche a impliqué de multiples ajustements d'entraînement et d'évaluation de modèles, chacun affinant la précision de la détection et minimisant les faux positifs.

Dans ce travail nous également développé une interface utilisateur graphique (GUI) intuitive qui permet aux utilisateurs de tester des images et des vidéos pour la détection des météores. Cette interface prend en charge diverses configurations de modèles et introduit une fonction de détection multi-événements, synchronisant plusieurs sources vidéo pour identifier les occurrences simultanées de météores.

Ce mémoire ouvre la voie à de futures avancées dans le domaine de la détection des météores, en proposant une solution pratique et déployable qui intègre la recherche théorique et l'application dans le monde réel. Les résultats démontrent des améliorations significatives dans la précision de la détection, ouvrant la voie à de nouvelles explorations dans ce domaine captivant.

Abtsract

The detection of meteors presents a unique blend of challenges and opportunities in the realm of astronomical observation. This dissertation embarks on an innovative journey to enhance meteor detection using advanced deep-learning techniques. We delved into the intricacies of meteoroids, meteors, and meteorites, exploring their scientific significance and the complexities involved in their detection.

Building on this foundation, we curated a robust dataset and experimented with cutting-edge models, including Faster RCNN with ResNet and Swin Transformer backbones. Our approach involved multiple iterations of model training and evaluation, each refining the detection accuracy and minimizing false positives.

We also developped of an intuitive Graphical User Interface (GUI) that allows users to test images and videos for meteor detection. This interface supports various model configurations and introduces a multievent detection feature, synchronizing multiple video sources to identify simultaneous meteor occurrences.

This thesis sets the stage for future advancements in meteor detection, offering a practical and deployable solution that integrates theoretical research with real-world application. The results demonstrate significant improvements in detection accuracy, paving the way for further exploration in this captivating field.

Bibliographie

[1] Digging into detectron 2. https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd/.

[2] Freeture a free software to capture meteors for fripon. https://adsabs.harvard.edu/pdf/2014pim4.conf...39A.

[3] Meteors : Light from comets and asteroids. https://link.springer.com/chapter/10.1007/978-3-030-38509-5_2.

[4] opencv. https://opencv.org/.

[5] Openvino documentation 2022.3. https://docs.openvino.ai/2022.3/home.html/.

[6] Overcoming the challenges of bringing ai to the edge. https://www.flir.com/discover/cores-components/Overcoming-the-Challenges-of-Bringing-AI-to-the-Edge/.

[7] python. https://www.python.org/.

[8] Region proposal network (rpn) in object detection. https://www.geeksforgeeks.org/region-proposal-network-rpn-in-object-detection/.

[9] Types of meteorites. https://www.sciencefacts.net/types-of-meteorites.html.

[10] Surabhi Agarwal, Elena Hervas-Martin, Jonathan Byrne, Aubrey Dunne, José Luis Espinosa Aranda, and David Rijlaarsdam. An evaluation of low-cost vision processors for efficient star identification. *Sensors*, 20 :6250, 11 2020.

[11] Meta AI. Pytorch.

[12] Aisha Al-Owais, Maryam E Sharif, Sarra Ghali, Maha Abu Serdaneh, Omar Belal, and Ilias Fernini. Meteor detection and localization using yolov3 and yolov4. *Neural Computing and Applications*, 35(21) :15709–15720, 2023.

[13] Simon Anghel, Dan A Nedelcu, Mirel Birlan, and Ioana Boaca. Single-station meteor detection filtering using machine learning on MOROI data. *Monthly Notices of the Royal Astronomical Society*, 518(2) :2810–2824, 11 2022.

[14] P. Beck, A. Pommerol, N. Thomas, B. Schmitt, F. Moynier, and J.-A. Barrat. Photometry of meteorites. *Icarus*, 218(1) :364–377, 2012.

[15] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4 : Optimal speed and accuracy of object detection. *arXiv preprint arXiv :2004.10934*, 2020.

[16] DEKALI Mounir BOUGHELOUS Imad. *Contribution à l'étude et à la réalisation d'un système pour la détection automatique et le suivi des météores lumineux*. 2023.

[17] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *Computer Vision–ECCV 2016 : 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 354–370. Springer, 2016.

[18] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[19] D. Cecil and M. Campbell-Brown. The application of convolutional neural networks to the automation of a meteor detection pipeline. *Planetary and Space Science*, 186 :104920, 2020.

[20] Chuan-Wang Chang, Santanu Santra, Jun-Wei Hsieh, Hendri Pirdiansyah, and Chi-Fang Lin. Multi-fusion feature pyramid for real-time hand detection. *Multimedia Tools and Applications*, 81, 04 2022.

[21] Changrui Chen, Yu Zhang, Qingxuan Lv, Shuo Wei, Xiaorui Wang, Xin Sun, and Junyu Dong. Rrnet : A hybrid detector for object detection in drone-captured images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[22] Gong Cheng, Jiabao Wang, Ke Li, Xingxing Xie, Chunbo Lang, Yanqing Yao, and Junwei Han. Anchor-free oriented proposal generator for object detection. *IEEE Transactions on Geoscience and Remote Sensing*, 60 :1–11, 2022.

[23] Florent Colas, Brigitte Zanda, Sylvain Bouley, Simon Jeanne, Adrien Malgoyre, Mirel Birlan, Cyrille Blanpain, Jérôme Gattacceca, Laurent Jorda, Julien Lecubin, et al. Fripon : a worldwide network to track incoming meteoroids. *Astronomy & Astrophysics*, 644 :A53, 2020.

[24] Marcelo De Cicco, Susana Zoghbi, Andres P. Stapper, Antonio J. Ordoñez, Jack Collison, Peter S. Gural, Siddha Ganju, Jose-Luis Galache, and Peter Jenniskens. Artificial intelligence techniques for automating the CAMS processing pipeline to direct the search for long-period comets. In Marc Gyssens and Jean-Louis Rault, editors, *International Meteor Conference, Petnica, Serbia*, pages 65–70, January 2018.

[25] NumPy Devs. Numpy.

[26] OpenVino Devs. Openvino.

[27] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88 :303–338, 2010.

[28] FRIPON. Fireball recovery and interplanetary observation network, 2024.

[29] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[30] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1) :142–158, 2015.

[31] Peter S Gural. Deep learning algorithms applied to the classification of video meteor detections. *Monthly Notices of the Royal Astronomical Society*, 489(4) :5109–5118, 09 2019.

[32] Gareth Halfacree. How cams, the cameras for allsky meteor surveillance project, detects long-period comets through machine learning, 2022.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] SnowFlake Inc. Streamlit.

[35] Jupyter. Jupyter notebooks.

[36] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. *arXiv preprint arXiv :1902.07296*, 2019.

[37] Pavel Koten, Karel Fliegel, Stanislav Vitek, and Petr Pata. Automatic video system for continuous monitoring of the meteor activity. *Earth Moon and Planets*, 108 :69–76, 05 2011.

[38] Alex Krizhevsky. Alexnet : Convolutional neural networks, 2012.

[39] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[40] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco : Common objects in context. In *Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[41] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer : Hierarchical vision transformer using shifted windows, 2021.

[42] Thiago Cesar Marsola, Ana Carolina Lorena, SP Itaú Unibanco, and SP Aeronautics. Meteor detection using deep convolutional neural networks. *Anais do Simp Bras de Autom Intel*, 1 :1042, 2019.

[43] Microsoft. Visual studio code.

[44] University of Western OntarioT. Somn fireball event of september 25, 2009, 2023.

[45] OpenMMlab. Mmcv.

[46] OpenMMlab. Mmdetection.

[47] Eloy Peña-Asensio, Josep M Trigo-Rodríguez, Pau Grèbol-Tomàs, David Regordosa-Avellana, and Albert Rimola. Deep machine learning for meteor monitoring : Advances with transfer learning and gradient-weighted class activation mapping. *Planetary and Space Science*, 238 :105802, 2023.

[48] Pillow. Pillow.

[49] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[50] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*, 2018.

[51] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[52] Rabea Sennlaub, Martin Hofmann, Mike Hankey, Mario Ennes, Thomas Müller, Peter Kroll, and Patrick Mäder. Object classification on video data of meteors and meteor-like phenomena : algorithm and data. *Monthly Notices of the Royal Astronomical Society*, 516(1) :811–823, 08 2022.

[53] E. Stempels and J. Kero. The Swedish Allsky Meteor Network : first results. In A. Roggemans and P. Roggemans, editors, *International Meteor Conference Egmond, the Netherlands, 2-5 June 2016*, page 288, January 2016.

[54] study.com. Meteorite | composition, classification types, 2023.

[55] UAA. Swedish allsky meteor network, 2023.

[56] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104 :154–171, 2013.

[57] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[58] Xiaobin Wang, Dekang Zhu, and Ye Yan. Towards efficient detection for small objects via attention-guided detection network and data augmentation. *Sensors*, 22(19) :7663, 2022.

[59] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[60] Chang Xu, Jinwang Wang, Wen Yang, and Lei Yu. Dot distance for tiny object detection in aerial images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1192–1201, 2021.

[61] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers : Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2129–2137, 2016.

[62] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[63] Xindi Zhang, Ebroul Izquierdo, and Krishna Chandramouli. Dense and small object detection in uav vision based on cascade network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[64] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr : Deformable transformers for end-to-end object detection. *arXiv preprint arXiv :2010.04159*, 2020.

Annexe

Le Code d'automatisation Python pour l'extraction et le téléchargement de vidéos du réseau suédois :

```
from google.colab import drive
drive.mount('/content/gdrive')
drive_folder_path = '/content/gdrive/My Drive/swedish_dataset'

import os
import requests
from bs4 import BeautifulSoup
# import certifi

ssl_cert_file = '/content/www.astro.uu.pem'

def download_files(folder_url):

    response = requests.get(folder_url, verify=False)

    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')

        image_links = soup.find_all('img', src=lambda src: src and 'comb.jpg' in src or 'centroid_xymap.png' in src)
        video_links = soup.find_all('a', href=lambda href: href and href.endswith('.avi'))

        common_name = folder_url.split('/')[-2]  # Extracting folder name from URL

        # Downloading images
        for i, image_link in enumerate(image_links):
            image_url = folder_url + image_link['src']
            image_filename = f"{common_name}_image_{i}.jpg"
            download_file(image_url, image_filename)

        # Downloading video
        for i, video_link in enumerate(video_links):
            video_url = folder_url + video_link['href']
            video_filename = f"{common_name}_video_{i}.avi"
            download_file(video_url, video_filename)
```

```
    else:
        print("Failed to retrieve webpage:", folder_url)

def download_file(file_url, filename):
    save_path = os.path.join(drive_folder_path, filename)
    response = requests.get(file_url, verify=False)

    if response.status_code == 200:
        with open(save_path, "wb") as file:
            file.write(response.content)
        print("Downloaded:", filename)
    else:
        print("Failed to download file:", file_url)
def scrape_and_download(base_url):
    response = requests.get(base_url, verify=False)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
        folder_links = soup.find_all('a', href=lambda href: href and href.
   endswith('/'))
        for folder_link in folder_links:
            print('folder_link:', folder_link)
            folder_url = base_url + folder_link['href']
            print('folder_url:', folder_url)
            download_files(folder_url)
    else:
        print("Failed to retrieve webpage:", base_url)

# base_url = "https://www.astro.uu.se/~meteor/UAA/data/trigger/2022/"
base_url = "https://www.astro.uu.se/~meteor/UAA/data/trigger/"

scrape_and_download(base_url)
```

Listing 1 – Code d'automatisation Python pour l'extraction et le téléchargement de vidéos du réseau suédois

Outils et Librairies utilisés :

- **BeautifulSoup** : Utilisé pour analyser les documents HTML et XML afin d'extraire les URLs.
  - `soup.find_all('img', src=lambda src: src and 'comb.jpg' in src or 'centroid_xymap.png' in src)` recherche tous les liens vers des images qui contiennent des sous-chaînes spécifiques dans leur attribut `src`.
  - `soup.find_all('a', href=lambda href: href and href.endswith('.avi'))` recherche tous les liens vidéo qui se terminent par `.avi`.
- **Certifi** : garantit la sécurité des requêtes HTTP en vérifiant les certificats SSL.
- **Requests** : Facilite les requêtes HTTP pour télécharger des fichiers multimédias.
  - `response = requests.get(folder_url, verify=False)` envoie une requête à l'URL donnée et récupère le contenu.
  - `response = requests.get(file_url, verify=False)` envoie une demande de téléchargement du fichier à partir de l'URL donnée.

0.0.0.1 Défis rencontrés

- **Swedish website :**
  - Gérer le contenu dynamique et les pages rendues par JavaScript.
  - Assurer la conformité avec les conditions de service du site web afin d'éviter les problèmes juridiques.
  - Gérer de grands volumes de données sans atteindre les limites de requêtes du serveur.
- **FRIPON :**
  - Les vidéos avec des espaces vides, où la majorité des événements déclarés n'étaient que des espaces vides qui ne contenaient pas d'événement météorologique.
  - La difficulté de télécharger les données manuellement, ce qui prend beaucoup de temps compte tenu des espaces vides, des événements non confirmés et des déclarations de faux positifs.
  - Le programme Freeture qui fonctionne avec FRIPON déclare beaucoup de faux positifs, par exemple en confondant les nuages, les avions et les insectes avec les météores.
- **Critère de Sélection :**
  - Résolution et qualité de l'image.
  - Étiquetage clair des météores et des non-météores.
  - Accessibilité légale et droits d'utilisation.

0.0.0.2 Vidéos Suédois Collectées

Les vidéos dans le lien suivant ont été collectées en scrappant les données du réseau suédois :

Vidéos Suédois Collectées

Ces vidéos ne sont pas de simples collections statiques ; elles détiennent la clé pour débloquer des insights pour notre modèle d'IA. Nous plongerons dans chaque image, en les extrayant et en les préparant méticuleusement pour créer une Dataset riche. Ce jeu de données servira ensuite de base pour entraîner notre IA, lui permettant d'apprendre et de faire des prédictions précises.

Le Code d'extraction de frames des videos :

```python
import cv2
import numpy as np
import os
import json

MIN_AREA = 5  # Minimum area threshold
MAX_AREA = 10000  # Maximum area threshold

def make_square(x, y, w, h):
    # Find the maximum side length (either width or height)
    side_length = max(w, h)

    # Calculate the new coordinates of the square bounding box
    new_x = x + (w - side_length) // 2
```

```
    new_y = y + (h - side_length) // 2
    new_w = side_length
    new_h = side_length

    return new_x, new_y, new_w, new_h

bg_subtractor = cv2.createBackgroundSubtractorMOG2(history=100,
    varThreshold=50, detectShadows=True)

# Define the folder containing the videos
video_folder = "vv/"
output_folder = "frames/"
box_folder = "box/"

# Create the output folder if it doesn't exist
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

if not os.path.exists(box_folder):
    os.makedirs(box_folder)

# List to store image filenames and bounding box coordinates
image_data = []

def get_video_list(folder_path):
    # Get list of all files in the folder
    all_files = os.listdir(folder_path)

    # Define video file extensions (modify as needed)
    video_extensions = [".avi", ".mp4", ".m4v"]  # Add more extensions if
    needed

    # Filter files based on extensions
    video_list = [f for f in all_files if f.lower().endswith(tuple(
    video_extensions))]

    return video_list

video_filenames = get_video_list(video_folder)

print(len(video_filenames))

# Loop through each video file
for filename in video_filenames:
    video_path = os.path.join(video_folder, filename)

    # Open the video
    cap = cv2.VideoCapture(video_path)

    # Check if video opened successfully
    if not cap.isOpened():
        print(f"Error opening video: {video_path}")
        continue

```

```
    # Get video properties (optional for output video)
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = int(cap.get(cv2.CAP_PROP_FPS))

    cpt = 0
    goodones = []
    coord = []

    # Process each frame in the video
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        fg_mask = bg_subtractor.apply(gray_frame)

        # Find contours
        contours, _ = cv2.findContours(fg_mask, cv2.RETR_EXTERNAL, cv2.
    CHAIN_APPROX_SIMPLE)
        if len(contours) == 0:
            continue
        max_contour_index = lambda contours: max(enumerate(contours), key=
    lambda x: cv2.contourArea(x[1]))
        # Calculate the index and area of the contour with the maximum
    area
        max_contour_index, max_area_contour = max_contour_index(contours)
        max_area = cv2.contourArea(max_area_contour)

        if MIN_AREA < max_area < MAX_AREA:
            x, y, w, h = cv2.boundingRect(max_area_contour)
            x = int(x - (w * 1))
            y = int(y - (h * 1))
            w = int(w + (w * 2))
            h = int(h + (h * 2))

            x, y, w, h = make_square(x, y, w, h)

            # Save the frame
            goodones.append((frame, frame.copy()))
            coord.append((x, y, w, h))
            # Draw bounding box (optional)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Release video capture resources
    cap.release()
    idx = 0
    for img, coord in zip(goodones, coord):
        img_filename = f"{filename}_frame_{idx}.jpg"
        img_path = os.path.join(output_folder, img_filename)
        box_img_path = os.path.join(box_folder, img_filename)
        cv2.imwrite(img_path, img[1])
        cv2.imwrite(box_img_path, img[0])
```

```

        # Store image filename and bounding box coordinates in the list
        image_data.append({
            "image_filename": img_filename,
            "bounding_box": {
                "x": coord[0],
                "y": coord[1],
                "width": coord[2],
                "height": coord[3]
            }
        })

        idx += 1

# Serialize image data to JSON
json_filename = "image_data_new.json"
json_path = os.path.join(output_folder, json_filename)
with open(json_path, "w") as json_file:
    json.dump(image_data, json_file, indent=4)

print("Images and bounding box data saved successfully.")
```

Listing 2 – Code Python pour l'extraction de frames des videos

```
model = dict(
    type='FasterRCNN',
    backbone=dict(
        type='SwinTransformer',
        depths=[2, 2, 6, 2],
        embed_dims=96,
        num_heads=[3, 6, 12, 24],
        mlp_ratio=4,
        qkv_bias=True,
        patch_norm=True,
        window_size=7,
        drop_rate=0.0,
        attn_drop_rate=0.0,
        drop_path_rate=0.2,
        convert_weights=True,
        with_cp=False,
        out_indices=(0, 1, 2, 3)),
    neck=dict(
        type='FPN',
        in_channels=[96, 192, 384, 768],
        out_channels=256,
        num_outs=5),
    rpn_head=dict(
        type='RPNHead',
        in_channels=256,
        feat_channels=256,
        anchor_generator=dict(
            type='AnchorGenerator',
            scales=[8],
            ratios=[0.5, 1.0, 2.0],
            strides=[4, 8, 16, 32, 64]),
```

```
32          bbox_coder=dict(
33              type='DeltaXYWHBBoxCoder',
34              target_means=[0.0, 0.0, 0.0, 0.0],
35              target_stds=[1.0, 1.0, 1.0, 1.0]),
36          loss_cls=dict(type='CrossEntropyLoss', use_sigmoid=True,
    loss_weight=1.0),
37          loss_bbox=dict(type='L1Loss', loss_weight=1.0)),
38      roi_head=dict(
39          type='StandardRoIHead',
40          bbox_roi_extractor=dict(
41              type='SingleRoIExtractor',
42              roi_layer=dict(type='RoIAlign', output_size=7, sampling_ratio
    =0),
43              out_channels=256,
44              featmap_strides=[4, 8, 16, 32]),
45          bbox_head=dict(
46              type='Shared2FCBBoxHead',
47              in_channels=256,
48              fc_out_channels=1024,
49              roi_feat_size=7,
50              num_classes=2,
51              bbox_coder=dict(
52                  type='DeltaXYWHBBoxCoder',
53                  target_means=[0.0, 0.0, 0.0, 0.0],
54                  target_stds=[0.1, 0.1, 0.2, 0.2]),
55              reg_class_agnostic=False,
56              loss_cls=dict(type='CrossEntropyLoss', use_sigmoid=False,
    loss_weight=1.0),
57              loss_bbox=dict(type='L1Loss', loss_weight=1.0)))
58      )
```

Listing 3 – Architecture Faster R-CNN et Swin backbone

0.1 Calibration

0.1.1 Calibration pour le suivi futur :

La détection des météores ne consiste pas seulement à capturer des instants fugaces d'éclat céleste ; il s'agit aussi de percer les mystères de ces voyageurs de l'espace. L'une des étapes cruciales de cette exploration est l'étalonnage méticuleux de notre caméra panoramique équipée d'un objectif fish-eye. Cette étape est la pierre angulaire des futures recherches visant à suivre les météores et à prédire les sites d'atterrissage des météorites, permettant ainsi une étude plus approfondie de ces visiteurs cosmiques.

0.1.2 The Processus de Calibration : de la 3D à la 2D

Notre caméra AllSky, avec son grand objectif fish-eye, offre une vue complète du ciel nocturne. Cependant, cette perspective panoramique s'accompagne de son lot de défis. L'objectif fish-eye déforme l'image, ce qui rend impératif le calibrage de la caméra pour traduire ces vues déformées en coordonnées 3D précises. Cet étalonnage permet de s'assurer que chaque

météore capturé n'est pas une simple traînée de lumière, mais un point précis dans le ciel. Le processus de calibration consiste à obtenir toutes les informations nécessaires (paramètres ou coefficients) sur la caméra afin de déterminer une relation précise entre un point 3D dans le monde réel et sa projection 2D correspondante (pixel) dans l'image capturée. Ce processus implique généralement la récupération de deux types de paramètres :

1. **Paramètres intrinsèques** : ils concernent le système caméra/objectif, y compris la longueur focale, le centre optique et les coefficients de distorsion radiale de l'objectif.
2. **Paramètres extrinsèques** : ils décrivent l'orientation (rotation et translation) de la caméra par rapport à un système de coordonnées mondiales.

Pour projeter un point 3D sur le plan de l'image, nous transformons d'abord le point du système de coordonnées du monde au système de coordonnées de la caméra à l'aide des paramètres extrinsèques (Rotation $\mathbf{R}$ et Translation $\mathbf{t}$)). Ensuite, en utilisant les paramètres intrinsèques de la caméra, nous projetons le point sur le plan de l'image.

L'objectif du processus de calibration est de trouver la 3*3 matrice intrinsèque $\mathbf{K}$, la 3*3 matrice de rotation $\mathbf{R}$ et le vecteur de translation 3×1 $\mathbf{t}$ en utilisant un ensemble de points 3D connus (X_w, Y_w, Z_w) et leurs coordonnées d'image correspondantes (u, v). Lorsque nous disposons des paramètres intrinsèques et extrinsèques, la caméra est considérée comme calibrée.

0.1.3 Entrées et Sorties de l'Algorithme Calibration de caméra

Entrées : Une collection d'images avec des points dont les coordonnées 2D de l'image et les coordonnées 3D du monde sont connues.

Outputs : La matrice intrinsèque 3×3 de la caméra, la rotation et la translation de chaque image.

Les équations reliant un point 3D (X_w, Y_w, Z_w) en coordonnées mondiales à sa projection (u, v) en coordonnées image sont :

$$\begin{bmatrix} u' \\ v' \\ z' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$u = \frac{u'}{w'}$$

$$v = \frac{v'}{w'}$$

$\mathbf{P}$ est une matrice de projection 3×4 composée de deux parties : la matrice intrinsèque $\mathbf{K}$ et la matrice extrinsèque $[\mathbf{R} \mid \mathbf{t}]$.

$$\mathbf{P} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \times \overbrace{[\mathbf{R} \mid \mathbf{t}]}^{\text{Extrinsic Matrix}}$$

Comme indiqué, la matrice intrinsèque **K** est triangulaire supérieure :

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

où,
- cx, cy sont les coordonnées x et y du centre optique dans le plan de l'image. - Dans notre cas, nous nous intéressons aux coefficients de distorsion
— f_x, f_y sont les distances focales x et y (généralement identiques).
— c_x, c_y sont les coordonnées x et y du centre optique dans le plan de l'image.
— γ est l'asymétrie entre les axes, généralement nulle.
Dans notre cas, nous nous intéressons aux **coefficients de distorsion**

Les coefficients de distorsion D tiennent compte des distorsions radiales et tangentielles de l'objectif fisheye. Pour une caméra fisheye, D est typiquement une matrice 4x1. Ces coefficients sont utilisés pour corriger l'image en compensant les distorsions de l'objectif.

Après avoir pris des photos du damier que nous utilisons pour calibration de Caméra AllSky (FIGURE 27), nous tenons le motif dans différentes positions et sous différents angles, car les motifs doivent apparaître déformés de différentes manières.

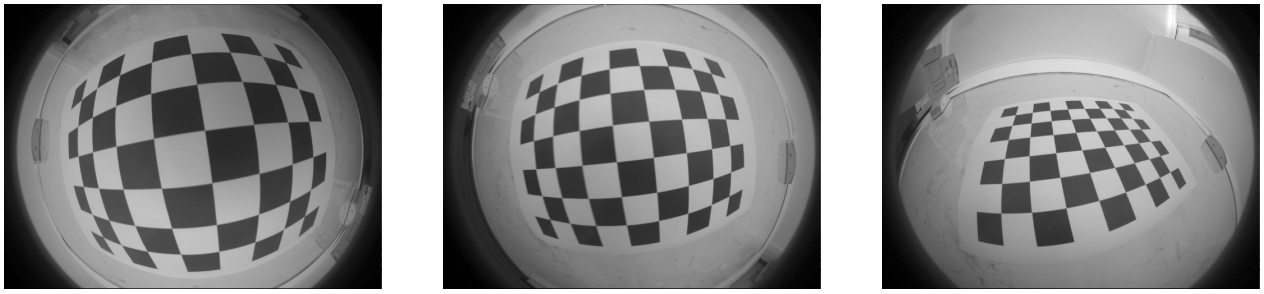

FIGURE 27 – Série d'images prises pour effectuer Calibration des caméras AllSky

Code de Calibration :

```
1 import cv2
```

```
assert cv2.__version__[0] == '3', 'The fisheye module requires opencv
    version >= 3.0.0'
import numpy as np
import os
import glob

CHECKERBOARD = (7,9)
subpix_criteria = (cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER, 30,
    0.1)
calibration_flags = cv2.fisheye.CALIB_RECOMPUTE_EXTRINSIC + \
                    cv2.fisheye.CALIB_CHECK_COND + \
                    cv2.fisheye.CALIB_FIX_SKEW

objp = np.zeros((1, CHECKERBOARD[0]*CHECKERBOARD[1], 3), np.float32)
objp[0,:,:2] = np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]].T.reshape
    (-1, 2)

_img_shape = None
objpoints = []  # 3d point in real world space
imgpoints = []  # 2d points in image plane.
images = glob.glob('*.jpg')

for fname in images:
    img = cv2.imread(fname)
    if _img_shape is None:
        _img_shape = img.shape[:2]
    else:
        assert _img_shape == img.shape[:2], "All images must share the
    same size."
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Find the chess board corners
    ret, corners = cv2.findChessboardCorners(gray, CHECKERBOARD, cv2.
    CALIB_CB_ADAPTIVE_THRESH + \
                                             cv2.CALIB_CB_FAST_CHECK + \
                                             cv2.CALIB_CB_NORMALIZE_IMAGE)
    # If found, add object points, image points (after refining them)
    if ret:
        objpoints.append(objp)
        cv2.cornerSubPix(gray, corners, (3,3), (-1,-1), subpix_criteria)
        imgpoints.append(corners)

N_OK = len(objpoints)
K = np.zeros((3, 3))
D = np.zeros((4, 1))
rvecs = [np.zeros((1, 1, 3), dtype=np.float64) for i in range(N_OK)]
tvecs = [np.zeros((1, 1, 3), dtype=np.float64) for i in range(N_OK)]

rms, _, _, _, _ = cv2.fisheye.calibrate(
    objpoints,
    imgpoints,
    gray.shape[::-1],
    K,
    D,
    rvecs,
```

```
    tvecs,
    calibration_flags,
    (cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER, 30, 1e-6)
)

print("Found " + str(N_OK) + " valid images for calibration")
print("DIM=" + str(_img_shape[::-1]))
print("K=np.array(" + str(K.tolist()) + ")")
print("D=np.array(" + str(D.tolist()) + ")")
```

Listing 4 – Code Calibration de Caméra

Code l'élimination de distorsion :

```
import cv2
import numpy as np
import sys

K = np.array([[fx, 0, cx], [0, fy, cy], [0, 0, 1]])  # Replace with actual values
D = np.array([[k1, k2, k3, k4]])  # Replace with actual distortion coefficients
DIM = (w, h)  # Replace with actual dimensions

def undistort(img_path):
    img = cv2.imread(img_path)
    h, w = img.shape[:2]
    map1, map2 = cv2.fisheye.initUndistortRectifyMap(K, D, np.eye(3), K, DIM, cv2.CV_16SC2)
    undistorted_img = cv2.remap(img, map1, map2, interpolation=cv2.INTER_LINEAR, borderMode=cv2.BORDER_CONSTANT)
    cv2.imshow("undistorted", undistorted_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

if __name__ == '__main__':
    for p in sys.argv[1:]:
        undistort(p)
```

Listing 5 – Elimination de Distorsion

0.1.4 Du ciel à la terre : Prévoir les sites d'atterrissage des météorites

Une fois la caméra calibrée, la véritable magie commence. Lorsque les météores traversent le ciel, plusieurs caméras enregistrent leur trajectoire. En utilisant les coordonnées calibrées de ces caméras, nous pouvons déterminer la trajectoire exacte des météores. Cela implique le calcul de la hauteur, de la vitesse et de l'angle de descente, ce qui fournit des données cruciales pour prédire l'endroit où la météorite pourrait atterrir.

Ce processus de triangulation s'apparente à un roman policier cosmique. De multiples caméras AllSky, placées à des endroits stratégiques, observent la descente du météore. Chaque caméra fournit une pièce du puzzle et, ensemble, elles révèlent le voyage du météore depuis

l'espace jusqu'à son lieu de repos final sur Terre. Ces informations sont précieuses pour les chercheurs qui souhaitent récupérer des météorites, car elles leur permettent de localiser les sites d'atterrissage potentiels avec une précision sans précédent.

Alors que nous continuons à affiner et à améliorer nos systèmes de détection, cette base calibrée soutiendra d'innombrables explorations futures, faisant du voyage de chaque météore une histoire de découverte, de science et d'émerveillement.