République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 - Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Système Informatique

Thème

Détection des Piétons par Apprentissage Profond de Transformateur

---

Présenté par :

CHIBI Moncef

Encadré Par :

Dr. BENCHERIET Chemmse Ennahar

Président :

Dr. HALLACI Samir

Examinateur :

Dr. MADI Laila

2022/2023

Remerciement

Tout d'abord, je tiens à exprimer ma gratitude envers Dieu Tout-Puissant de m'avoir accordé la volonté et la persévérance nécessaires pour mener à bien ce travail.

Je saisis cette occasion pour exprimer mes sincères remerciements et ma reconnaissance à Mme. Chemmse Ennahar Benchriet, ma directrice de mémoire, pour son précieux soutien et ses conseils tout au long de mes recherches.

J'aimerais également remercier les membres du jury pour l'intérêt qu'ils ont porté à mes travaux en acceptant d'examiner mon travail et en contribuant à son enrichissement par leurs suggestions.

Je souhaite exprimer ma profonde reconnaissance envers tous les responsables et enseignants du département d'informatique de l'Université de Guelma qui ont contribué à ma formation.

Je tiens également à exprimer ma gratitude envers la directrice et l'ingénieur du Laboratoire d'Automatique et d'Informatique de Guelma (LAIG).

Enfin, j'adresse mes sincères remerciements à tous mes proches et amis qui m'ont toujours encouragé lors de la préparation de ce mémoire.

Dédicace

Je dédie ce travail à

Mes parents, qui m'ont élevé, enseigné, encouragé et soutenu tout au long de ma vie.

Mes chers frères, sans qui ma vie n'aurait aucun sens.

Tous les membres de ma famille, mes amis et collègues.

Tous mes enseignants.

Et à tous qui m'ont aide de près ou de loin pour la réalisation de ce travail.

Moncef

Résumé

La securite routiere et autoroutiere est devenue un probleme de haute priorite a cause de l'augmentation massive des accidents de la route par annee.

Dans le cadre de notre projet, notre objectif est de concevoir une application permettant de détecter les piétons dans les zones routières. Pour cela, Nous avons combiné la puissance des Transformateurs et des Réseaux de Neurones Convolutifs (CNN) pour obtenir une détection précise et rapide des piétons. Le modèle DETR (Detection Transformer) a été utilisé, car il intègre à la fois des CNN pour extraire les caractéristiques des images d'entrée et des transformateurs pour former le modèle.Nous avons ré-entrainer modèle DETR pour detecter uniqument les piétons. DETR-P est une adaptation du modèle DETR pour se concentrer uniquement sur la détection des piétons. Nous avons utilisé l'ensemble de données PRW pour tester et évaluer notre système.les performances de notre système ont été testées sur des images et des vidéos de complexité variable. Les résultats obtenus sont très promoteurs mais peuvent être améliorés.

Mots clés : Détection des piétons, Apprentissage profond, Transformateur, CNN, DETR.

Abstract

Road and highway safety has become a high-priority issue due to the significant increase in road accidents per year.

In the scope of our project, our objective is to design an application that detects pedestrians in road areas. To achieve this, we have combined the power of Transformers and Convolutional Neural Networks (CNNs) to obtain accurate and fast pedestrian detection. We utilized the DETR (Detection Transformer) model because it integrates both CNNs for extracting features from input images and transformers for model training. We retrained the DETR model specifically for pedestrian detection, resulting in DETR-P, an adaptation of the DETR model focused solely on pedestrian detection. We used the PRW dataset to test and evaluate our system. The performance of our system has been tested on images and videos of varying complexity. The obtained results are very promising but can be improved.

Keywords : Pedestrian detection, Deep learning, Transformer, CNN, DETR.

ملخص

أصبحت السلامة على الطرق والطرق السريعة قضية ذات أولوية عالية بسبب الزيادة الكبيرة في حوادث الطرق سنويًا. في نطاق مشروعنا، هدفنا هو تصميم تطبيق يكتشف المشاة في مناطق الطرق. لتحقيق ذلك، قمنا بدمج قوة المحولات والشبكات العصبية التلافيفية (CNN) للحصول على اكتشاف دقيق وسريع للمشاة. استخدمنا نموذج) DETR محول الاكتشاف) لأنه يدمج كلاً من شبكات CNN لاستخراج الميزات من صور الإدخال والمحولات لتدريب النموذج. أعدنا تدريب نموذج DETR خصيصًا لاكتشاف المشاة، مما أدى إلىDETR-P ، وهو تعديل لنموذج DETR يركز فقط على اكتشاف المشاة. استخدمنا مجموعة بيانات PRW لاختبار وتقييم نظامنا. تم اختبار أداء نظامنا على الصور ومقاطع الفيديو متفاوتة التعقيد. النتائج التي تم الحصول عليها واعدة للغاية ولكن يمكن تحسينها بشكل أكبر.

الكلمات المفتاحية: كشف المشاة، التعلم العميق، المحولات، CNN، DETR.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

En raison de l'augmentation rapide des accidents de la route causant un nombre important de victimes chaque année, les chercheurs dans le domaine des Systèmes de Transport Intelligent (STI) se sont engagés à développer des applications visant à améliorer la sécurité routière. Parmi les principaux défis à relever figurent la détection précoce des obstacles, la reconnaissance des panneaux de signalisation et la surveillance des comportements des véhicules suspects pouvant provoquer des accidents.

Les accidents les plus meurtriers sont souvent le résultat de situations critiques telles que la traversée de la route par des piétons ou des animaux. Un exemple tragique est le carambolage survenu sur l'autoroute Est-Ouest à Ain Defla en 2021, où 34 véhicules ont été impliqués, entraînant la perte de cinq vies et faisant douze blessés. Cet accident s'est produit lorsque le conducteur d'un véhicule en panne a été percuté par un autre véhicule lors de sa tentative de traversée de la route.

Dans le cadre de ce projet, notre objectif est de concevoir une application novatrice permettant de détecter les piétons dans les zones routières, afin d'alerter les conducteurs à temps pour qu'ils puissent prendre les mesures de sécurité appropriées, telles que ralentir, freiner ou éviter une collision. Pour cela, nous combinons l'apprentissage profond des transformateurs visuels (Visual Transformer) avec les capacités des réseaux de neurones convolutifs (CNN) pour une détection précise et rapide des piétons.

Notre mémoire se compose de trois chapitres qui abordent de manière approfondie les différents aspects de notre projet :

- Chapitre I présente une étude détaillée sur les systèmes de transport intelligent, en soulignant l'importance de la détection des piétons dans la prévention des accidents. Nous examinons également l'état de l'art en matière de détection des piétons, en mettant en évidence quelques recherches récentes dans ce domaine.

- Chapitre II est consacré à une étude approfondie sur les réseaux de neurones convolutifs et les transformateurs visuels. Nous explorons les principes fondamentaux de ces deux approches et analysons leurs forces et leurs faiblesses respectives.
- Dans le Chapitre III, nous détaillons la conception et l'architecture du système proposé. Nous expliquons comment les transformateurs visuels et les réseaux de neurones convolutifs sont combinés pour améliorer la détection des piétons dans les zones routières. Nous présentons également les principaux résultats obtenus au cours de nos expérimentations ainsi que leur interpritation.

Et nous cloturons cette etude par une conclusion générale et des prospectives pour des travaux futurs qui seront elabore par d'autre etudiants.

Chapitre 1

Détection des piétons

1 Introduction

Au cours des dernières années un accroissement rapide des accidents de la route a été noté. Les accidents les plus meurtriers ont été causés par la traversé de la route des animaux ou des piétons, l'exemple le plus terrifiant c'est le carambolage de 34 véhicules sur l'autoroute Est-Ouest à Ain Defla en 2021 causant au moins 5 morts et 12 blessés et qui à été causé par un automobiliste en panne qui s'est fait percuter par une voiture en voulant traverser la route. Avec le grand développement dans le domaine de l'intelligence artificielle ce phénomène peut être réduit en créant un system pour la détection des piétons dans les zones routières afin d'alerter le conducteur a temps pour prendre ses précautions sécuritaires (ralentissement, freinage...etc.)

2 Système de transport intelligent

Un système de transport intelligent (ITS) est l'application des technologies de détection, d'analyse, de contrôle et de communication au transport terrestre afin d'améliorer la sécurité, la mobilité et l'efficacité. Les STI comprennent une large gamme d'applications qui traitent et partagent des informations pour réduire les embouteillages, améliorer la gestion du trafic, minimiser l'impact environnemental et augmenter les avantages du transport pour les utilisateurs commerciaux et le public en général. [w1]

3 Détection d'objets

La détection d'objets est une technique de vision par ordinateur permettant de localiser des instances d'objets dans des images ou des vidéos. Les algorithmes de détection d'objets tirent généralement parti de l'apprentissage automatique ou de l'apprentissage en profondeur pour produire des résultats significatifs. Le but de la détection d'objets est de reproduire cette intelligence à l'aide d'un ordinateur. Il existe plusieurs types de détection d'objets, par exemple la détection de visage ou la détection de personne. La détection de la présence humaine est un cas particulier de détection d'objet, où l'on cherche à détecter la présence et la localisation précise, dans une image, d'une ou plusieurs personnes, en général dans une posture proche de celle de la station debout ou de la marche. [w2]

4 Détection de personnes

La détection de personnes est la localisation de toutes les personnes présentes sur une image, vidéo par l'intelligence artificielle, tout comme la détection d'objet, l'algorithme détecte l'objet dans l'image ou la vidéo. Il le sépare de l'arrière-plan et enferme l'objet détecté dans un rectangle. La détection humaine est une forme très difficile de détection d'objet. C'est parce que les personnes se présentent sous différentes formes et couleurs. Les personnes peuvent également se déplacer de différentes manières. Pour que l'algorithme puisse détecter les personnes en mouvement, il doit apprendre tous ces mouvements possibles, et doit être alimentés avec une énorme quantité de données visuelles. [w3]

5 Applications de détection de personnes

Les applications de détection de personnes peuvent servir à de nombreuses fins tel que la détection des piétons , l'analyse des mouvements de personnes,la compréhension du comportement humain, la sécurité du travail, l'identification de personnes ,la sécurité et la surveillance...etc.

6 Detection des piétons

La détection des piétons est une technologie clé derrière les systèmes avancés d'aide à la conduite (ADAS) qui permettent aux voitures d'effectuer la détection des piétons pour améliorer la sécurité routière. La détection des piétons est également utile dans des applications telles que la vidéosurveillance ou les systèmes de récupération d'images. La détection des piétons est composée principalement des modules illustrés par la figure 1.1 . [w4]

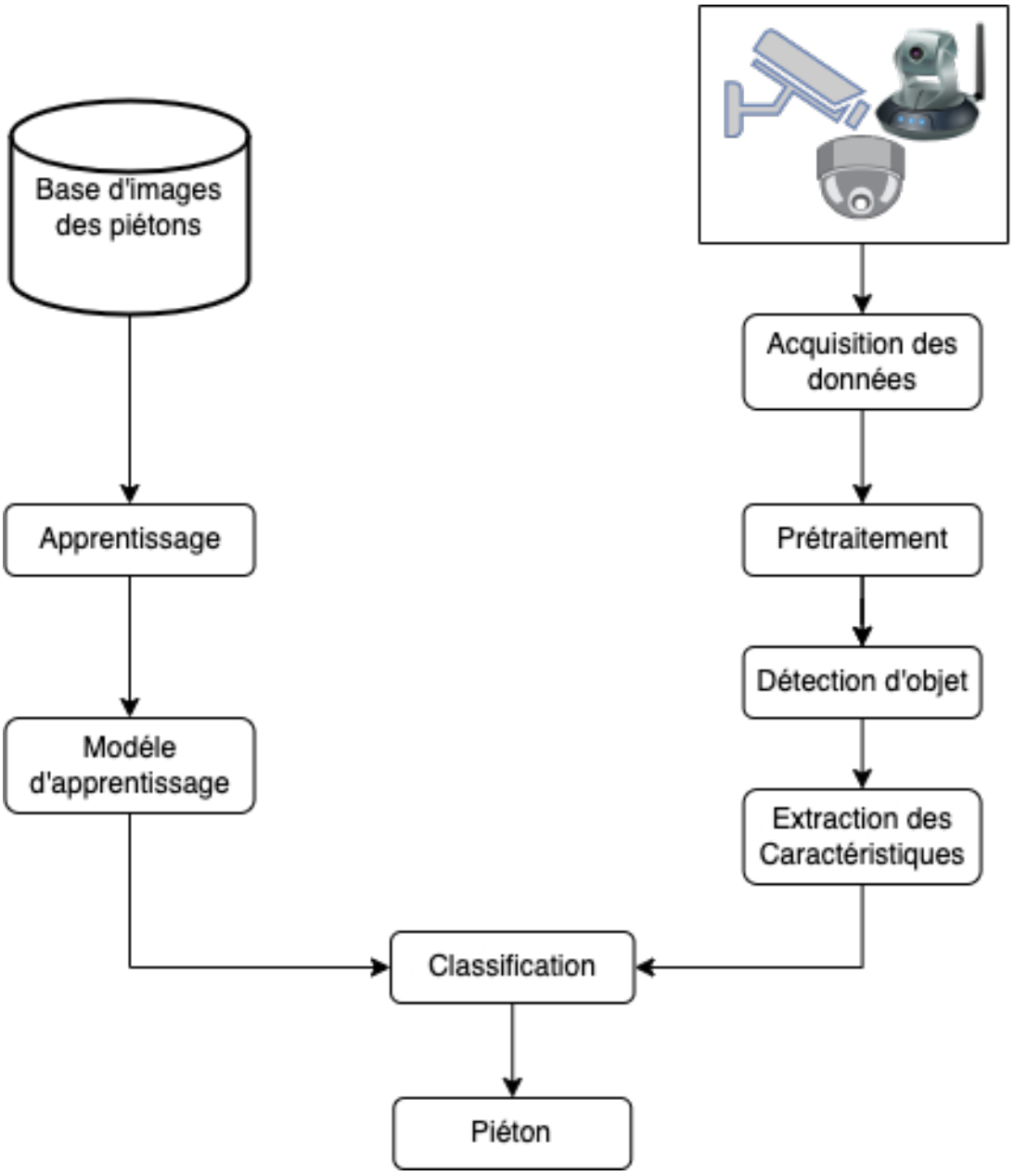


FIGURE 1.1 – Schéma général d'un systéme de détection des piétons

7 Base d'images des piétons

7.1 Base 'Caltech Pedestrian' [1]

L'ensemble de données sur les piétons Caltech se compose d'environ 10 heures de vidéo 640x480 30Hz prises à partir d'un véhicule circulant dans un trafic régulier dans un environnement urbain. Environ 250 000 images (en 137 segments d'environ une minute) avec un total de 350 000 boîtes englobantes et 2 300 piétons uniques ont été annotées. L'annotation comprend une correspondance temporelle entre les boîtes englobantes et les étiquettes d'occlusion détaillées.

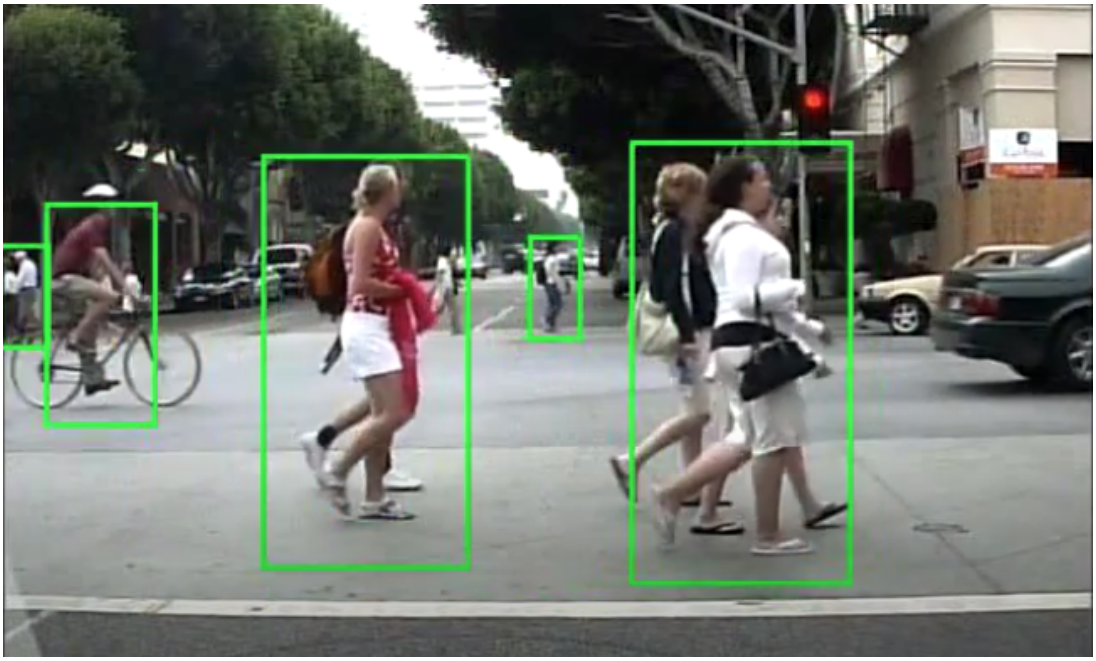


FIGURE 1.2 – Caltech pedestrian dataset

7.2 Base 'Daimler' [2]

L'ensemble de données contient 15560 échantillons de piétons (découpes d'images à une résolution de 48×96) et 6744 images complètes supplémentaires sans piétons pour extraire des échantillons négatifs. L'ensemble de test contient une séquence indépendante avec plus de 21790 images et 56492 étiquettes de piétons (entièrement visibles ou partiellement masquées), capturées à partir d'un véhicule pendant 27 minutes de conduite dans le trafic urbain.

FIGURE 1.3 – Daimler dataset

7.3 Base 'CrowdHuman' [3]

CrowdHuman est un vaste ensemble de données de détection humaine richement annoté, qui contient respectivement 15 000, 4 370 et 5 000 images collectées sur Internet pour la formation, la validation et les tests.

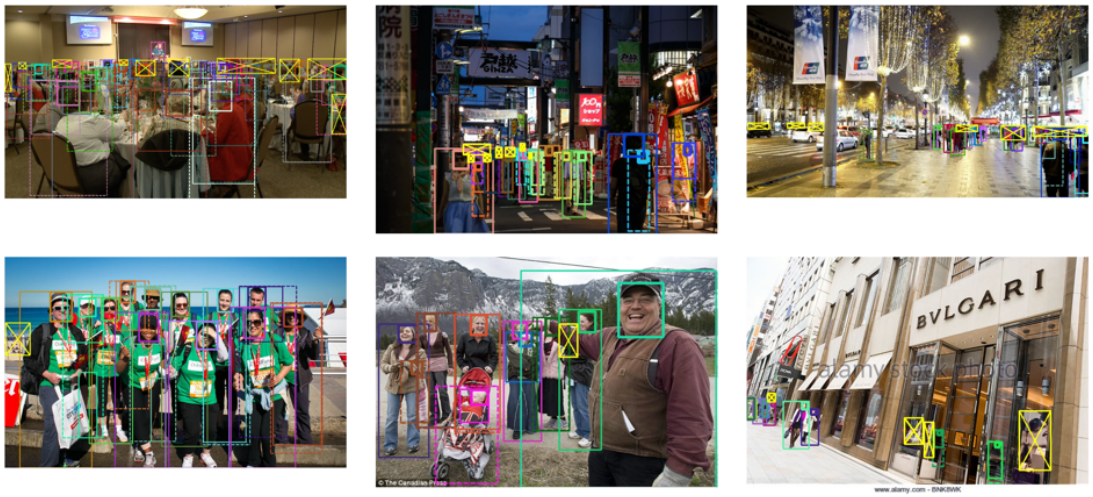


FIGURE 1.4 – CrowdHuman dataset

7.4 Base 'EuroCity Persons' [4]

L'ensemble de données EuroCity Persons fournit un grand nombre d'annotations très diverses, précises et détaillées de piétons, cyclistes et autres cyclistes dans des scènes de circulation urbaine. Les images de cet ensemble de données ont été collectées à bord d'un

véhicule en mouvement dans 31 villes de 12 pays européens. Avec plus de 238 200 instances de personnes étiquetées manuellement dans plus de 47 300 images.

FIGURE 1.5 – EuroCity Persons dataset

7.5 Base 'Kaist' [5]

L'ensemble de données se compose de 95 000 paires couleur-thermique (640 x 480, 20 Hz) prises à partir d'un véhicule. Tous les couples sont annotés manuellement pour un total de 103 128 annotations denses et 1 182 piétons uniques.

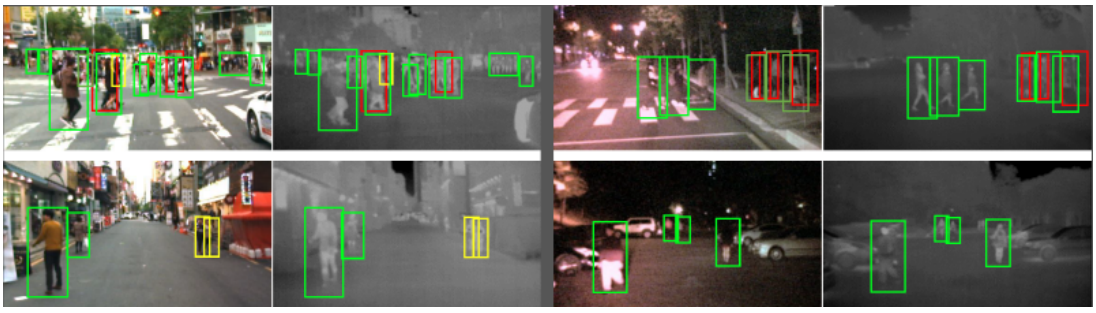


FIGURE 1.6 – Kaist dataset

7.6 Base 'CroHD' [6]

L'ensemble de données CroHD enregistré dans des scénarios encombrés se compose de 4 vidéos de formation correspondant à 5740 trames de formation et de 5 vidéos de test correspondant à 5723 trames de test. La densité moyenne de piétons sur l'ensemble des vidéos est d'environ 178 piétons par image.

FIGURE 1.7 – CroHD dataset

7.7 Base 'PRW'

La base 'PRW (Person Re-identification in the Wild) Dataset[11]'. Cette base contient 2 dossiers principaux. Le dossier "frames", il contient 11816 images, et le dossier "annotations" contient 11816 fichiers mat en correspondance avec chaque image dans le dossier "frames". Toutes les cases annotées sont des piétons. Chaque fichier MAT enregistre la position de la boîte englobante dans le cadre.

FIGURE 1.8 – PRW dataset

8 Quelques travaux sur la détections des piétons

Dans le cadre de notre projet nous avons fait l'étude de quelques travaux récents de la littérature scientifique afin de positionner notre travail.

8.1 Article 1 : Pedestrian Head Detection and Tracking via Global Vision Transformer - Détection et suivi de la tête des piétons via Transformateur de vision globale [6]

Résume

Dans cet article les auteurs proposent une méthode pour détecter et suivre les têtes de piétons dans des scènes bondées. Tout d'abord, ils intègrent les transformateurs dans le "Tracker", qui apprend les relations entre les requêtes d'objets et les caractéristiques globales de l'image pour raisonner sur les résultats de détection dans chaque image, et fait également correspondre les requêtes d'objets et suit les objets entre les images adjacentes pour effectuer l'association de données. Deuxièmement, le Tracker proposé est évalué sur l'ensemble de données CroHD. Cet ensemble de données enregistré dans des scénarios encombrés se compose de 4 vidéos de formation correspondant à 5740 trames de formation et de 5 vidéos de test correspondant à 5723 trames de test. La densité moyenne de piétons sur l'ensemble des vidéos est d'environ 178 piétons par image. L'évaluation des performances de suivi sont mesurées par des métriques standards : "Multiple Object Tracking Accuracy (MOTA)", "the ratio of correctly identified detections (IDF1)",et "Higher Order Tracking Accuracy (HOTA)" . Les mesures supplémentaires utilisées pour évaluer tous les aspects du PHDTT proposé sont MT 'Mostly tracked targets', ML 'Mostly lost targets', FP 'false positives' , FN 'false negatives' et ID Sw 'identity switches'.Comme le montre le tableau I.1, la méthode PHDTT proposé surpasse tous les trackers de l'état de l'art par une nette marge. Plus précisément, la méthode proposée atteint 60,6 MOTA qui surpasse SORT par 14,2 MOTA, V_IOU par 7,2 MOTA, Tracktor par 1,7 MOTA ,et HeadHunter par 2,8 MOTA. Il vérifie l'efficacité et la capacité de généralisation du "Tracker PHDTT". La méthode proposée joint les tâches de détection et d'association dans le modèle unique de bout en bout selon lequel chaque tâche peut aider à apprendre efficacement une autre tâche .La méthode proposée fonctionne mal sur les vidéos qui contiennent la plus forte

densité des piétons, le flou de mouvement et le problème d'occlusion de la foule.

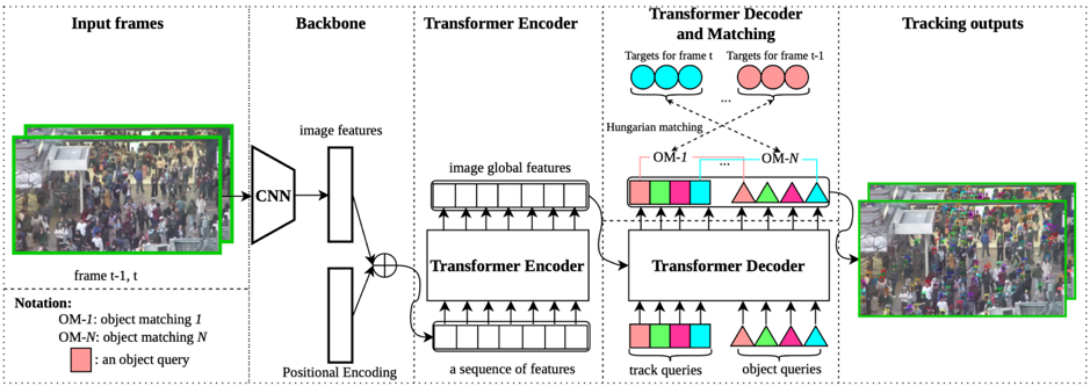


FIGURE 1.9 – Architecture du systéme proposé (PHDTT) [6]

Method	MOTA	IDF1	IDEucl	HOTA	MT	ML	FP	FN	ID Sw.
SORT	46.4	48.4	58.0	-	2.1	9.2	-	-	**649**
V_IOU	53.4	35.4	34.3	-	3.4	7.8	-	-	1,890
Tracktor	58.9	38.5	31.8	-	5.3	**5.0**	-	-	3,474
HeadHunter	57.8	**53.9**	**54.2**	**36.8**	31.9	19.9	**51,840**	**299,459**	4,394
Our PHDTT	**60.6**	47.9	52.6	36.1	**47.1**	8.0	132,714	184,215	15,004

TABLE 1.1 – Comparaison avec l'etat de l'art des méthodes de suivi sur l'ensemble de test CroHD [6]

8.2 Article 2 : Bispectral Pedestrian Detection Augmented with Saliency Maps using Transformer - Détection des piétons augmentée avec saillance Cartes utilisant Transformateur [7]

Résume

Dans cet article, les auteurs se concentrent sur le problème de la détection automatique des piétons pour les applications de surveillance. En particulier, l'objectif principal est d'effectuer une détection en temps réel à partir de caméras visibles et thermiques pour des aspects complémentaires. ils étudient l'utilisation d'images visibles et thermiques bispectrales pour la détection des piétons dans diverses conditions d'éclairage. En particulier, ils utilisent un schéma de fusion qui utilise des caractéristiques d'images appariées visibles et thermiques avec leurs cartes de saillance profonde correspondantes. Les canaux d'image

visible et thermique sont censés être complémentaires, où les images visibles ont tendance à fournir des détails de couleur et de texture, tandis que les images thermiques sont sensibles à la température de l'objet, ce qui peut être très utile dans différentes conditions d'éclairage. l'approche proposée est développée par l'utilisation de YOLOv3 comme architecture de base pour construire un réseau neuronal profond pour effectuer la détection dans des applications en temps réel. L'approche proposée est évaluée sur KAIST qui se compose d'environ 95k cadres sur l'environnement de trafic urbain et d'annotations denses pour 1182 piétons différents. Il est divisé en un ensemble d'apprentissage de 50,2 k images, et un ensemble de test de 45,1 k images. le modèle personnalisé YOLOv3 a été implémenté sur framework Darknet mais avec quelques configurations et modifications afin d'adapter le modèle a l'ensmeble de données KAIST. l'efficacité de l'architecture proposée est prouvée par l'obtention de meilleurs résultats quantitatifs et qualitatifs par rapport aux entrées simples et aux autres schémas de fusion. Comme le montre le tableau 1.2, l'approche proposée obtient les meilleurs résultats, 75,8 % en termes de mAP (Mean Average Precision), avec une marge de 17.7 % et 9.9 % pour les entrées visibles et thermiques, respectivement. Et une marge de 13%, 15.4%, 16.4%, 13.7%, 12.4%, 13.4% pour les autres schémas de fusion (input, early, halfway2, halfway3, halfway4, et late), respectivement.

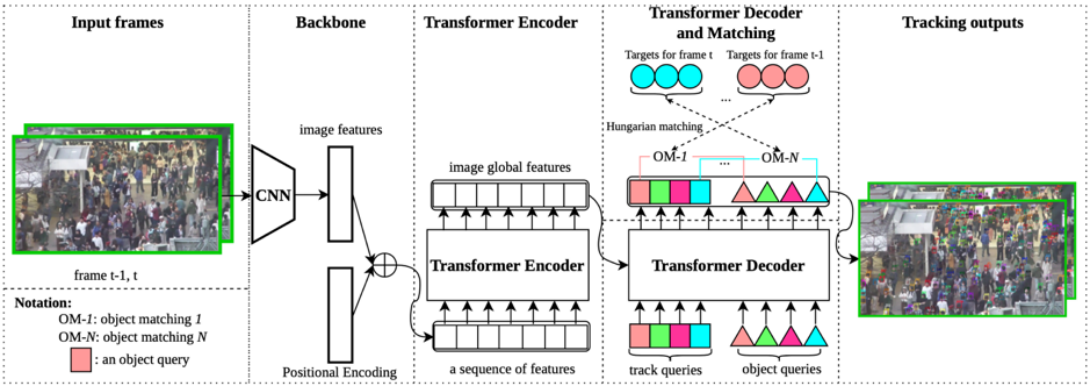


FIGURE 1.10 – Architecture du systéme proposé (PHDTT) [6]

Methods	Day	Night	All
Visible input	64.4	42.3	58.1
Thermal input	62.7	72.6	65.9
Input-fusion	69.2	46.0	62.8
Early-fusion	67.8	42.1	60.4
Halfway fusion-2	68.5	39.2	59.4
Halfway fusion-3	69.6	44.0	62.1
Halfway fusion-4	69.3	49.5	63.4
Late-fusion	67.7	48.8	62.4
Ours	72.6	79.0	**75.8**

TABLE 1.2 – Comparaison des performances de détection à l'aide d'une seule entrée et de différents niveaux de fusion[7]

8.3 Article 3 : Improved Pedestrian Attribute Recognition Using Swin Transformer and Semantic Self-Attention - amélioration de la reconnaissance des attributs des piétons à l'aide de Swin Transformer et de l'auto-attention sémantique [8]

Résume

Cet article propose un nouveau réseau d'encodeurs-décodeurs pour la reconnaissance des attributs des piétons, appelé "Swin Transformer and Décoder for Pedestrian attribut recognition Network (STDP-Net)". Tout d'abord, les chercheurs utilisent un transformateur Swin qui utilise 'Self-attention' comme encodeur. Cela permet à la méthode proposée de comprendre la relation relative entre les régions spatiales des images, contrairement aux méthodes conventionnelles basées sur la convolution. Cela permet une reconnaissance précise des attributs, même dans les entrées d'images de piétons mal alignées. après, ils ajoutent un décodeur pour comprendre les relations sémantiques entre les attributs. puis ils analysent comment chaque encodeur et décodeur affecte l'amélioration des performances. Ils ont utilisé six ensembles de données de référence PAR : PETA, PA100K , RAP , RAP2 , PETAzs et RAP2zs pour évaluer la méthode proposée.Le tableau 1.3 présente les résultats de la méthode proposée et des approches de pointe sur les ensembles de données PETA, PA100K et RAP.Le tableau 1.4 présente les résultats pour les ensembles de données RAP2, PETAzs et RAP2zs.La méthode proposée a obtenu les meilleurs résultats pour la plupart des métriques utilisées.

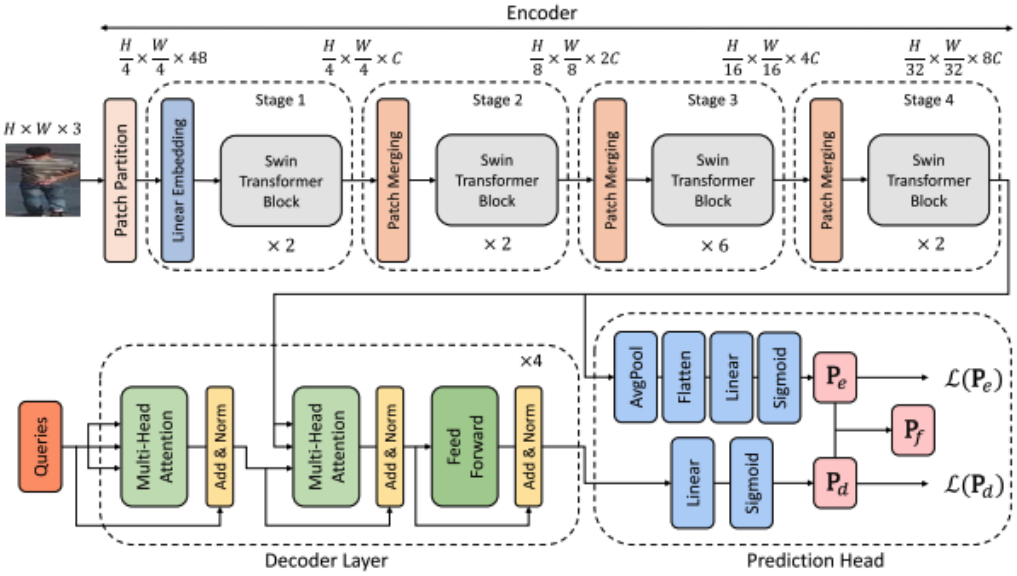


FIGURE 1.11 – Architecture du systéme proposé (STDP-Net)

Method	PETA						PA100K						RAP					
	mA	Acc	Prec	Recall	F1	Avg.	mA	Acc	Prec	Recall	F1	Avg.	mA	Acc	Prec	Recall	F1	Avg.
DeepMAR	82.89	75.07	83.68	83.14	83.41	81.64	72.70	70.39	82.24	80.42	81.32	77.41	73.79	62.02	74.92	76.21	75.56	72.50
HPNet	81.77	76.13	84.92	83.24	84.07	82.03	74.21	72.19	82.97	82.09	82.53	78.99	76.12	65.39	77.33	78.79	78.05	75.14
JRL	85.67	-	86.03	85.34	85.42	-	-	-	-	-	-	-	77.81	-	78.11	78.98	78.58	-
LGNet	-	-	-	-	-	-	76.96	75.55	86.99	83.17	85.04	81.54	78.68	68.00	80.36	79.82	80.09	77.55
PGDM	82.97	78.08	86.86	84.68	85.76	83.67	74.95	73.08	84.36	82.24	83.29	79.58	74.31	64.57	**78.86**	75.90	77.35	74.20
ResNet50	84.12	79.07	87.30	85.94	86.61	84.61	79.27	78.35	**87.25**	86.54	86.89	83.66	79.70	68.49	78.48	82.70	80.53	77.98
ALM	86.30	79.52	85.65	**88.09**	86.85	85.28	80.68	77.08	84.21	88.84	86.46	83.45	81.87	68.17	74.71	86.48	80.16	78.29
ALM	84.78	78.68	85.63	87.09	86.35	84.51	78.87	77.45	85.92	86.72	86.32	83.06	77.95	66.18	76.48	81.18	78.76	76.11
SSC_{soft}	**86.52**	78.95	86.02	87.12	86.99	85.12	**81.87**	78.89	85.98	**89.10**	86.87	**84.54**	**82.77**	68.37	75.05	**87.49**	80.43	78.82
SSC_{hard}	85.92	78.53	86.31	86.23	85.96	84.59	81.02	78.42	86.39	87.55	86.55	83.99	82.14	68.16	77.87	82.88	79.87	78.18
SSC_{soft}	83.90	78.60	86.16	86.59	86.38	84.33	79.51	78.49	86.80	87.30	87.05	83.83	79.95	68.64	78.50	82.89	80.64	78.12
SSC_{soft}	83.87	78.33	86.54	85.76	86.16	84.13	79.96	78.68	86.84	87.56	87.20	84.05	79.52	68.43	78.34	82.81	80.51	77.92
STDP	85.98	**80.54**	**87.38**	87.70	**87.54**	**85.83**	80.44	**79.35**	87.01	88.08	**87.54**	84.48	82.08	**70.20**	78.37	85.49	**81.77**	**79.58**

TABLE 1.3 – Comparaisons des résultats pour les ensembles de données PETA, PA100K et RAP.

Method	RAP2						$PETA_{zs}$						$RAP2_{zs}$					
	mA	Acc	Prec	Recall	F1	Avg.	mA	Acc	Prec	Recall	F1	Avg.	mA	Acc	Prec	Recall	F1	Avg.
ResNet50	78.56	67.60	76.76	83.32	79.90	77.23	69.57	57.58	72.66	69.60	71.09	68.10	71.98	64.60	76.57	78.70	77.62	73.89
Jia *et al.*	79.19	65.27	76.26	79.89	78.03	75.73	71.62	58.19	**73.09**	70.33	71.68	68.98	72.32	63.61	76.88	76.62	76.75	73.24
ALM	75.07	67.46	**81.00**	78.22	79.58	76.27	70.86	58.72	73.04	71.52	72.28	69.28	70.78	64.25	**77.57**	76.94	77.25	73.36
SSC_{soft}	78.54	67.78	76.89	83.45	80.04	77.34	70.70	58.12	71.68	71.69	71.69	68.78	71.96	65.11	76.94	79.14	78.02	74.23
SSC_{soft}	78.45	67.58	76.61	83.50	79.91	77.21	70.04	57.73	71.08	71.80	71.43	68.42	71.69	64.65	76.59	78.78	77.67	73.88
STDP	**80.20**	**68.74**	76.94	**84.90**	**80.73**	**78.30**	**72.47**	**60.25**	71.80	**75.43**	**73.57**	**70.70**	**75.19**	**66.31**	75.46	**82.89**	**79.00**	**75.77**

TABLE 1.4 – Comparisons des résultats sur les ensembles de données RAP2, PETAzs et RAP2zs.

8.4 Article 4 : Pedestrian and Vehicle Detection Algorithm Based on Swin Transformer in Rainy Scenes - Algorithme de détection de piétons et de véhicules Basé sur Swin Transformateur dans les scènes pluvieuses [9]

Résume

Pour améliorer les performances de détection d'objets dans des scènes pluvieuses, un algorithme de détection de piétons et de véhicules de bout en bout (PVformer) avec module de drainage a été proposé, basé sur le transformateur Swin, qui améliore la qualité d'image et la précision de détection dans les scènes de pluie. Sur la base des blocs de transformateur, un modèle de mappage de caractéristiques à 4 branches a été introduit pour réaliser le drainage à partir d'une seule image, atténuant ainsi l'influence de l'occlusion des traînées de pluie sur les performances du détecteur. Selon le problème de détection de petits objets uniquement par transformateur visuel, ils ont conçu un bloc de perception d'amélioration locale basé sur CNN et transformateur. De plus, le modèle de drainage et le modèle de détection ont été combinés pour entraîner le modèle PVformer par apprentissage par transfert. Pour la tâche de drainage d'image, ils ont utilisé Rain100L, Rain100H[12] , Rain800[13]. Pour L'ensemble de données de détection des piétons et des véhicules les jours de pluie, ils ont utilisé RIDRIS[14] , KITTI [15], UA-DETRAC [16], COCO [17], PASCAL VOC. Pour vérifier l'efficacité de l'algorithme, ils comparent les résultats expérimentaux avec les modèles de détection les plus avancés :SSD,Yolov5,Faster-RCNN, Mobilenet [18], Swin-Transformer . Pour l'effet de détection les jours de pluie, du fait des contraintes visuelles, le taux de rappel du Mobilenet et du SSD était très faible. La plupart des objets n'ont pas pu être détectés, en particulier les piétons et les véhicules à petite échelle. Yolov5 et Faster-RCNN étaient légèrement meilleurs que les deux premiers algorithmes, mais le rapport de détection manquée et fausse était encore important. La précision et le taux de rappel de la version B de Swin Transformer (Swin-B) étaient significativement meilleurs que les quatre méthodes précédentes. PVformer était meilleur les jours de pluie et peut détecter plus précisément. Pour la précision de détection, les tests sur l'ensemble de test Rain-PV ont montré que la précision de détection de l'algorithme dans cet article était significativement plus élevée que celle basée sur le CNN.

Il a encore mieux fonctionné sur l'ensemble de test Real Rain-150 lors de vrais jours de pluie, avec une précision améliorée de 26,2 %. Cela était dû à l'utilisation de l'attention du transformateur pour capturer les informations contextuelles globales, établissant ainsi une dépendance à l'objet à longue portée et extrayant des caractéristiques d'image plus puissantes. De plus, en ajoutant le module de drainage d'image, la précision de détection du transformateur Swin a été améliorée de 4,2% sur l'ensemble de données Rain-PV et de 7,1 % sur l'ensemble de test Real Rain-150, respectivement.En termes de vitesse de détection, le temps de détection du PVformer pour une seule image était de 70,4 ms. Le temps de détection n'est pas aussi bon que les méthodes basées sur CNN. La raison en est que la quantité d'informations dans les images est beaucoup plus importante que celle des données textuelles, il est donc nécessaire de concevoir une structure Transformer plus adaptée à l'image pour réduire le coût de calcul.

Test Set	Model	Backbone	mAP/%	Time/ms
Rain-PV	Mobilenet	MobileNet-224	49.4	48.8
Rain-PV	SSD	VGG-16	51.6	31.3
Rain-PV	Yolov5	Yolov5x	67.4	36.3
Rain-PV	Faster-RCNN	ResNet-101	73.0	59.4
Rain-PV	Swin-Transformer	Swin-T	84.7	65.3
Rain-PV	Ours	Derain-PVformer	88.9	70.4
Real Rain-150	Mobilenet	MobileNet-224	37.2	48.8
Real Rain-150	SSD	VGG-16	39.6	31.3
Real Rain-150	Yolov5	Yolov5x	51.7	36.3
Real Rain-150	Faster-RCNN	ResNet-101	56.1	59.4
Real Rain-150	Swin-Transformer	Swin-T	75.2	65.3
Real Rain-150	Ours	Derain-PVformer	82.3	70.4

TABLE 1.5 – Les résultats de comparaison des performances des différents algorithmes de détection [9]

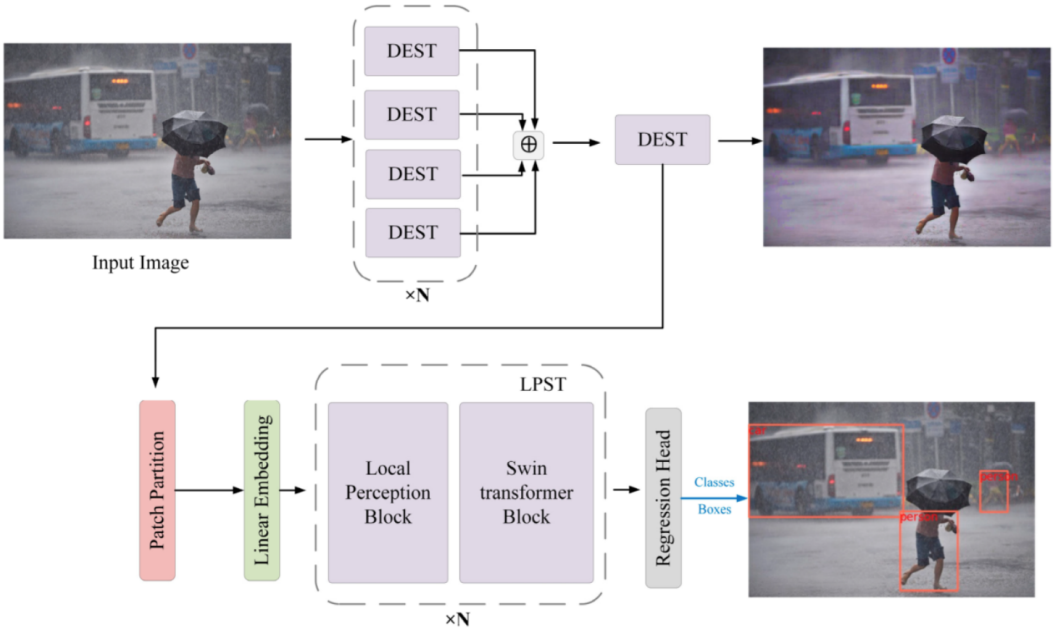


FIGURE 1.12 – Architecture du systéme proposé

8.5 Article 5 : Pedestrian Detection Using R-CNN Object Detector - Détection des piétons à l'aide du détecteur d'objets R-CNN [10]

Résume

Les techniques d'apprentissage en profondeur dans la détection des piétons ont démontré des résultats puissants dans des expériences et des recherches récentes. Dans cet article, L'expérience implique l'utilisation d'un modèle d'extraction de caractéristiques d'apprentissage en profondeur avec le détecteur R-CNN, qui est formé sur le modèle Alexnet, et évalué sur deux ensembles de données de détection de piétons différents qui sont Penn-Fudan et KTH Football. Les ensembles de données utilisé contiennent respectivement 170 images et 771 images. Ils utilisent 60 % de l'ensemble de données comme échantillon d'apprentissage pour l'ensemble de données Penn-Fudan et l'ensemble de données KTH Football. Les 40% restants des échantillons sont utilisés pour tester le détecteur R-CNN. Le détecteur R-CNN entraîné sur l'ensemble de données Penn-Fudan atteint une précision moyenne de 52 % et un taux d'échec moyen de 68 %. Le détecteur R-CNN entraîné sur l'ensemble de données KTH Football atteint une précision moyenne de 84 % et un taux d'échec moyen de 22 %. Les résultats obtenu démontre que la taille des données joue un

rôle important dans la détermination des performances du détecteur de piétons. Donc les détecteurs d'apprentissage en profondeur doivent être pris en charge par un ensemble de données à grande échelle pour améliorer les performances.

9 Conclusion

Dans ce chapitre, nous avons examiné le concept de système de détection des piétons, ainsi que certains ensembles de données utilisés dans ce domaine. De plus, nous avons également passé en revue quelques travaux de recherche sur la détection des piétons. L'objectif principal de ce chapitre était de fournir une base solide de connaissances sur la détection des piétons, en explorant les travaux existants et en identifiant les approches les plus pertinentes pour notre projet.

Chapitre 2

CNN vs Transformateur de Vision

1 Introduction

Bien que les réseaux de neurones convolutifs (CNN) aient été l'approche dominante pour le traitement d'images et les tâches de vision par ordinateur pendant de nombreuses années, l'architecture de transformateur visuel est récemment apparue comme une nouvelle technique prometteuse. L'idée clé derrière les transformateurs visuels est d'utiliser des mécanismes d'auto-attention pour apprendre des caractéristiques pertinentes de l'image en entrée, plutôt que de se fier aux couches de convolution comme dans les CNN. Au travers ce chapitre, nous allons présenter l'architecture des transformateur Visual (ViT) et CNN , comment ils fonctionnent, et la différence entre eux.

2 Réseaux de neurons Artificiels

Les réseaux de neurones sont une technique d'apprentissage automatique inspirée du fonctionnement du cerveau humain. Ils sont utilisés pour résoudre des problèmes complexes de classification, de prédiction et de reconnaissance de modèles dans des données. Les réseaux de neurones sont composés de plusieurs couches de neurones interconnectées, qui reçoivent des entrées et effectuent des calculs pour produire des sorties. L'apprentissage se produit en ajustant les poids des connexions entre les neurones pour minimiser une fonction de coût, qui mesure l'écart entre les prédictions du modèle et les vraies valeurs. Les réseaux de neurones peuvent être entraînés à l'aide de divers algorithmes d'optimisation,

tels que la descente de gradient, et sont utilisés dans de nombreux domaines, tels que la reconnaissance de la parole, la vision par ordinateur, la prédiction de séries chronologiques, etc.[19]

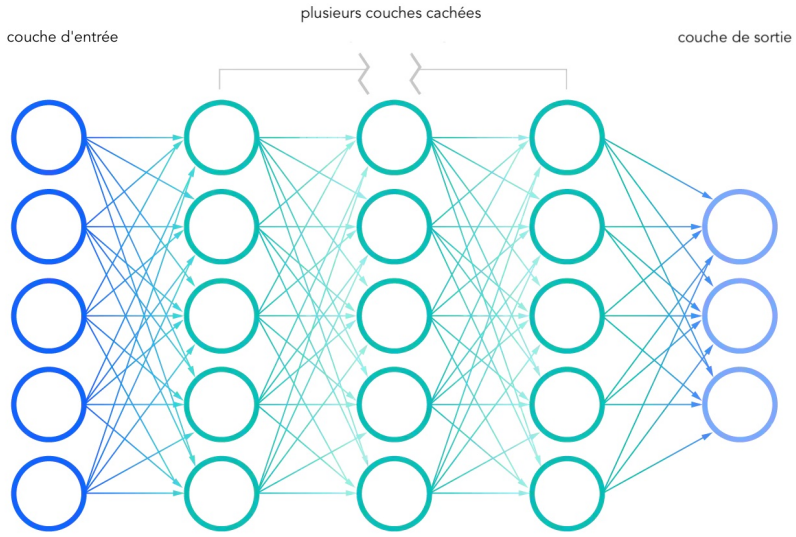


FIGURE 2.1 – Schéma d'un réseau de neurones artificiel

3 L'apprentissage automatique et l'apprentisage profond 'Machine learning and deep learning'

L'apprentissage profond est un sous-domaine de l'apprentissage automatique, qui est lui-même un sous-domaine de l'intelligence artificielle (IA). L'objectif central de l'IA est de fournir un ensemble d'algorithmes et de techniques qui peuvent être utilisés pour résoudre des problèmes que les humains effectuent de manière intuitive et presque automatique, mais qui sont par ailleurs très difficiles pour les ordinateurs. Un excellent exemple de cette catégorie de problèmes d'IA est l'interprétation et la compréhension du contenu d'une image - cette tâche est quelque chose qu'un humain peut faire avec peu ou pas d'effort, mais s'est avérée extrêmement difficile à accomplir pour les machines. Alors que l'IA vise à résoudre une gamme importante et diversifiée de problèmes liés au raisonnement automatique de la machine (inférence, planification, heuristique, etc.), le sous-champ de l'apprentissage automatique se concentre principalement sur la reconnaissance de modèles et l'apprentissage à partir de données.

3.1 Apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle (IA) qui se concentre sur la création de systèmes informatiques qui peuvent être formés pour effectuer des tâches basées sur des exemples, sans avoir besoin d'être explicitement programmés.[w6]

3.2 Apprentissage profond

L'apprentissage en profondeur est la dernière évolution de l'apprentissage automatique. Il utilise des algorithmes inspirés de la perception de notre cerveau pour enseigner aux ordinateurs comment nous apprenons, ce qui se rapproche le plus de l'imitation de l'intelligence humaine. Les réseaux de neurones à plusieurs couches sont appelés "apprentissage en profondeur", également parfois appelés "réseaux de neurones profonds".[w6]

3.3 Différence entre 'Machine Learning' et 'Deep Learning'

Alors que les algorithmes traditionnels d'apprentissage automatique ont une structure plutôt simple, telle que la régression linéaire ou les arbres de décision, l'apprentissage en profondeur est basé sur un réseau de neurones artificiels (ANN). Les algorithmes d'apprentissage en profondeur nécessitent de grandes quantités de données pour bien fonctionner. Étant donné que les algorithmes d'apprentissage automatique classiques peuvent apprendre à partir de moins d'exemples que ceux d'apprentissage en profondeur, ils sont beaucoup plus utiles pour les petites tâches. En raison de sa structure multicouche complexe, un système d'apprentissage en profondeur a besoin d'une grande quantité de données pour éliminer les fluctuations et faire prédictions précises.[w6]

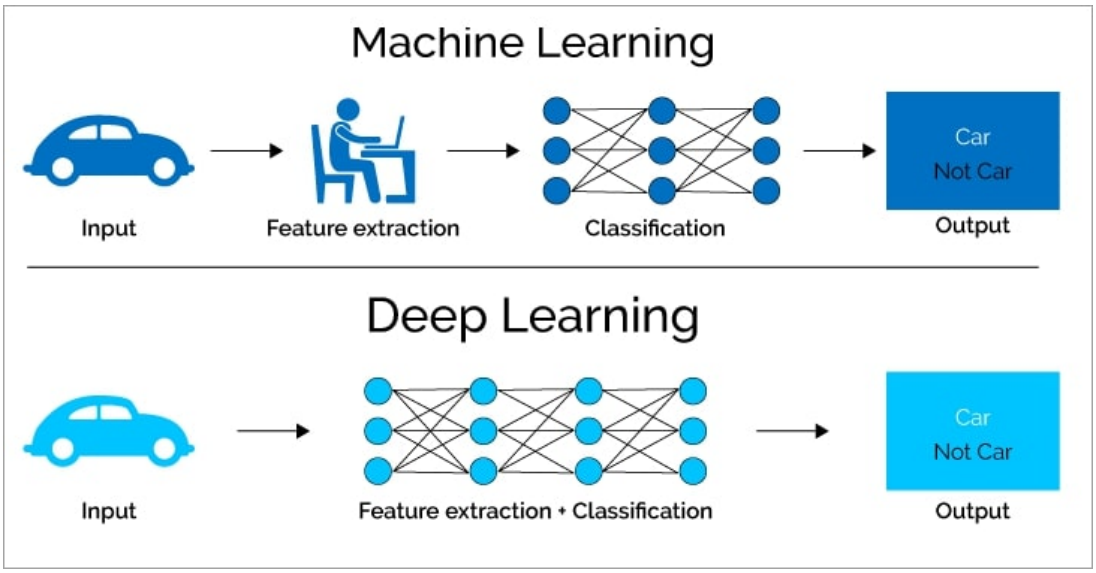


FIGURE 2.2 – Apprentissage automatique et Apprentissage profond

4 Les différents types d'apprentissage

Les méthodes d'apprentissage peuvent être classées en plusieurs catégories : apprentissage supervisé, apprentissage semi-supervisé, apprentissage non-supervisé et apprentissage par renforcement.

4.1 Apprentissage supervisé

L'apprentissage supervisé est une forme d'apprentissage automatique qui s'appuie sur des ensembles de données étiquetés. Ces ensembles de données sont conçus pour entraîner des algorithmes à classer les informations ou à prédire les résultats avec précision en mesurant leur précision et en s'améliorant au fil du temps. Il existe plusieurs méthodes d'apprentissage supervisé comme SVM, Adaboost, réseau de neurons. . .etc.[w7]

4.2 Apprentissage non-supervisé

L'apprentissage non supervisé utilise des algorithmes d'apprentissage automatique pour analyser et regrouper des ensembles de données non étiquetés. Ces algorithmes découvrent des modèles cachés dans les données sans intervention humaine.[w7]

4.3 Apprentissage semi-supervisé

Ce terme regroupe des méthodes qui se situent entre l'apprentissage non-supervisé et l'apprentissage supervisé. Ce type de méthodes est utilisé quand un grand nombre de données est disponible mais sans qu'elles soient toutes étiquetées. L'initialisation de la méthode est faite à partir d'un ensemble de données correctement étiquetées. Puis l'algorithme doit lui-même étiqueter les exemples suivants et construire son propre modèle. Plusieurs techniques d'apprentissage semi-supervisé existent parmi lesquels l'autoapprentissage, Co-apprentissage.

4.4 Apprentissage par renforcement

L'apprentissage par renforcement est une branche de l'apprentissage automatique qui ne repose pas sur un ensemble de données statiques, mais opère dans un environnement dynamique et apprend des expériences collectées. Les données, ou expériences, sont collectées pendant l'entrainment par le biais d'interactions d'essais et d'erreurs entre l'environnement et un agent logiciel. Cet aspect est important, car il atténue le besoin de collecte de données, de pré-traitement et d'étiquetage avant l'entrainment. Concrètement, cela signifie qu'avec la bonne incitation, un modèle d'apprentissage par renforcement peut commencer à apprendre un comportement par lui-même, sans supervision (humaine).[w8]

5 Réseau de neurones à convolution (CNN)

Un réseau de neurones à convolution (CNN) est une architecture de réseau de neurones profonds qui est largement utilisée pour la reconnaissance d'images, la classification et la segmentation d'images. Les CNN sont particulièrement efficaces pour extraire des caractéristiques à partir d'images en utilisant des couches où chaque couche, utilisant une banque de noyaux convolutifs, effectue plusieurs transformations.

L'opération de convolution est utilisée pour extraire des caractéristiques utiles à partir de points de données corrélés localement. Cette méthode permet d'appliquer des noyaux convolutifs qui produisent une sortie qui est ensuite traitée par une unité de traitement non linéaire, également appelée fonction d'activation. Cette fonction d'activation permet

non seulement d'apprendre des abstractions, mais elle intègre également de la non-linéarité dans l'espace des fonctionnalités. Cette non-linéarité génère des modèles d'activations différents pour différentes réponses, ce qui facilite l'apprentissage des différences sémantiques dans les images. [20]

6 Structure globale d'un CNN

La caractéristique clé des réseaux de neurones à convolution (CNN) est leur capacité à exploiter la corrélation spatiale ou temporelle des données. La topologie d'un CNN est divisée en plusieurs étapes d'apprentissage, comprenant une combinaison de couches de convolution, d'unités de traitement non linéaires et de couches de sous-échantillonnage.

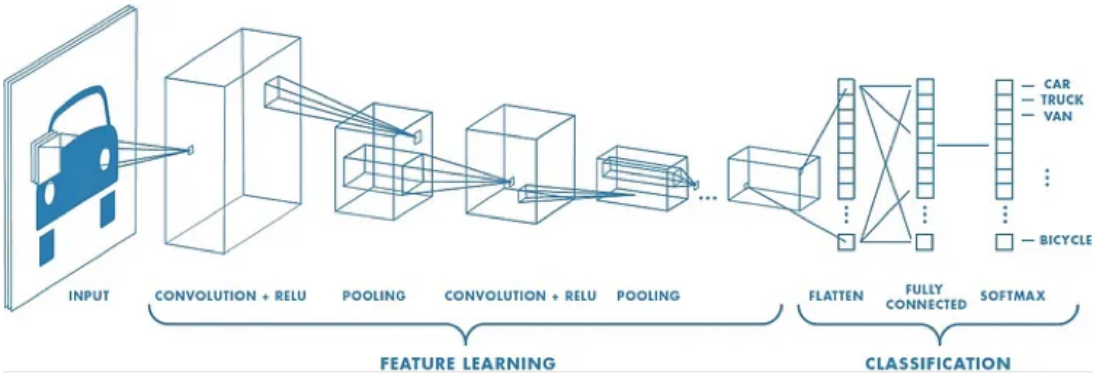


FIGURE 2.3 – Réseau de neurones avec de nombreuses couches convolutives

6.1 Couche de convolution

Une couche de convolution est une couche de traitement dans un réseau de neurones convolutif (CNN) qui effectue une opération de convolution sur les entrées. La convolution est une opération mathématique qui combine deux fonctions pour produire une troisième fonction qui représente la manière dont la forme d'une fonction est modifiée par l'autre fonction. Dans le contexte des réseaux de neurones, la convolution est utilisée pour extraire des caractéristiques des entrées qui sont importantes pour la tâche à accomplir, comme la reconnaissance d'images. Plus précisément, une couche de convolution applique un ensemble de filtres (ou noyaux) à une entrée pour produire une carte de caractéristiques. Chaque filtre est une matrice de poids qui glisse sur l'entrée et effectue une multiplication point à point avec la partie de l'entrée qui est actuellement couverte. Le résultat de chaque

multiplication est sommé pour donner une valeur unique pour chaque position du filtre sur l'entrée. Ce processus est répété pour chaque position de l'entrée, produisant une carte de caractéristiques qui met en évidence les motifs intéressants dans l'entrée. Les couches de convolution sont souvent utilisées dans les réseaux de neurones convolutifs pour extraire des caractéristiques à partir d'images, mais elles peuvent également être utilisées pour d'autres types d'entrées, comme le traitement de séquences.[w9][w10]

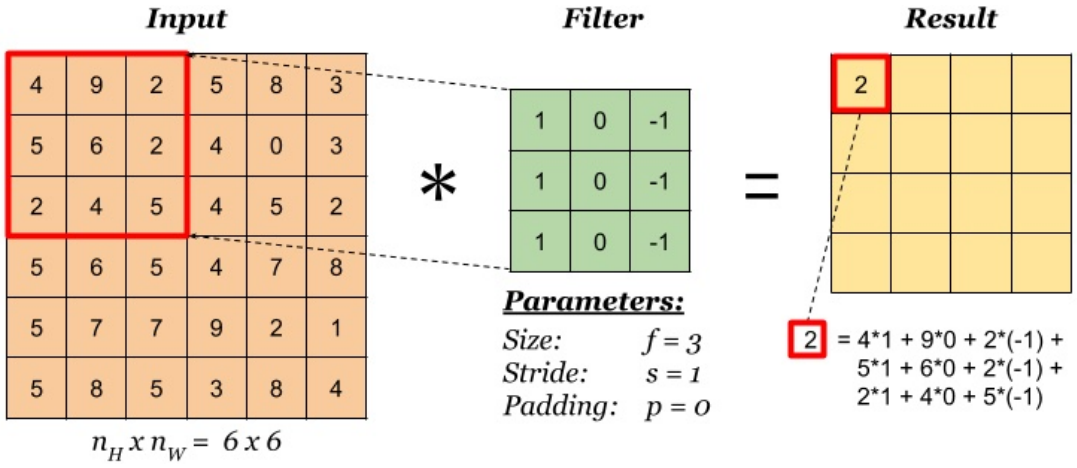


FIGURE 2.4 – Expemle explicative sur l'opération de convolution

6.1.1 Paramètres du filtre

La couche convolutionnelle contient des filtres pour lesquels il est important de savoir comment ajuster ses paramètres.[w9][w10]

6.1.2 Le pas 'Stride'

C'est le nombre de pixels décalés sur la matrice d'entrée. Lorsque le pas est de 1, nous déplaçons les filtres sur 1 pixel à la fois. Lorsque le pas est de 2, nous déplaçons les filtres sur 2 pixels à la fois et ainsi de suite.

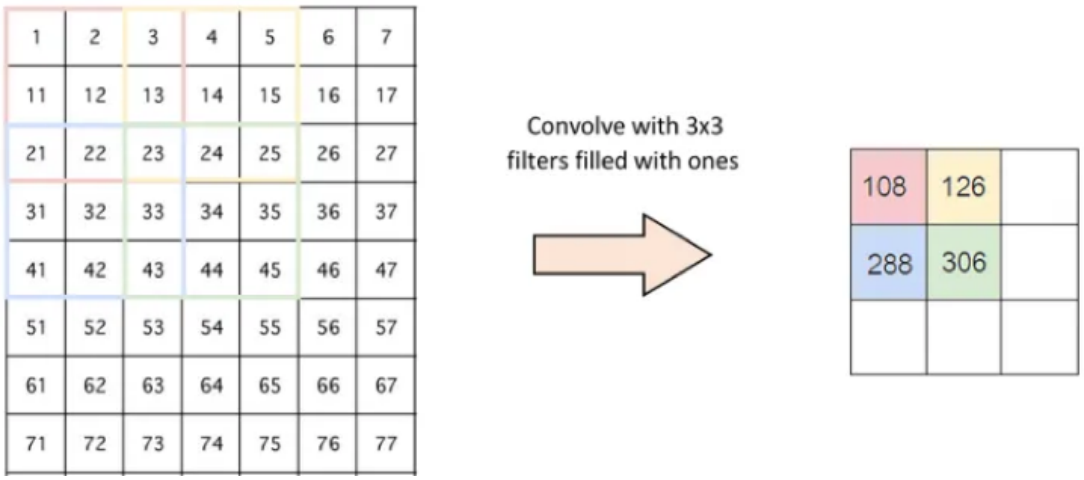


FIGURE 2.5 – Illustration du pas de 2 pixels

6.1.3 La marge à zéros 'Padding'

Parfois, le filtre ne correspond pas parfaitement à l'image d'entrée, pour résoudre ce problème, nous remplissons l'image avec des zéros (zéro-padding). Cette valeur peut être spécifiée soit manuellement, soit automatiquement par le biais d'une des configurations.

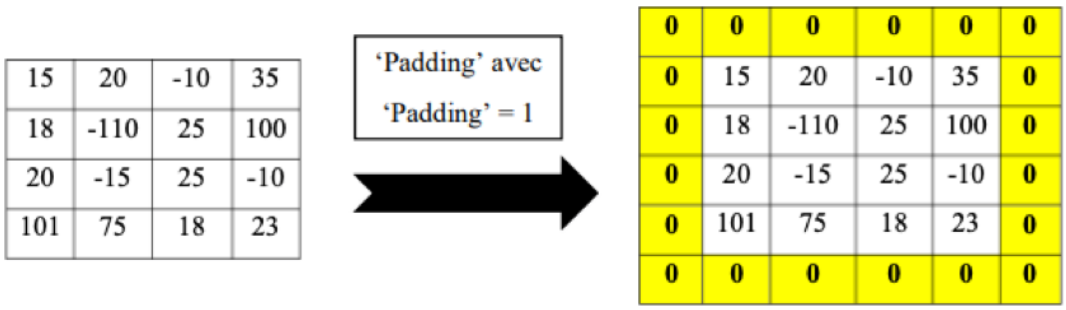


FIGURE 2.6 – Illustration de la marge à zéros

6.1.4 Dimensions d'un filtre

Un filtre de taille FxF appliqué à une entrée contenant C canaux est un volume de taille FxFxC qui effectue des convolutions sur une entrée de taille IxIxC et qui produit une carte de caractéristiques de sortie (aussi appelé activation map) de taille OxOx1 (avec un seul filtre). Appliquer K filtres de taille FxF engendre une carte de caractéristique de sortie de taille OxOxK.

6.2 Couche de pooling

La couche de pooling est généralement placée après une couche de convolution et applique une opération de sous-échantillonnage à la sortie de la couche de convolution. L'opération de sous-échantillonnage consiste à regrouper les activations voisines en une seule activation en utilisant une fonction d'agrégation telle que la moyenne ou le maximum. Cela permet de réduire la taille de la représentation en conservant les informations les plus importantes tout en réduisant le nombre de paramètres dans le modèle. Cela permet également de rendre le modèle plus robuste aux translations et aux petites variations de l'image d'entrée. Il existe plusieurs types de couche de pooling, notamment le max-pooling, le average-pooling et le global-pooling. Le max-pooling est le plus couramment utilisé car il a tendance à mieux fonctionner pour la plupart des applications de vision par ordinateur.[w9][w10]

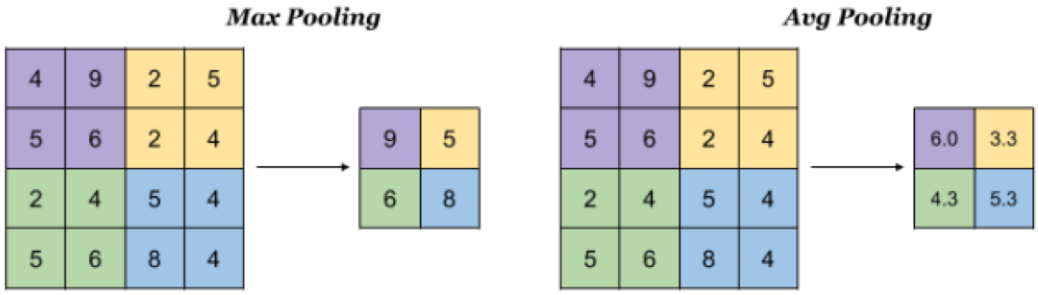


FIGURE 2.7 – Illustration du types de couche de pooling

6.3 Couche fully-connected

Une couche fully-connected (FC), également connue sous le nom de couche dense, est une couche de réseau de neurones où tous les neurones de la couche précédente sont connectés à tous les neurones de la couche suivante les couches fully-connected peuvent avoir un grand nombre de paramètres, ce qui peut entraîner un surapprentissage et des difficultés lors de l'entraînement de grands modèles. Pour cette raison, des architectures de réseaux de neurones plus récentes, comme les réseaux de neurones à attention ou les réseaux de neurones transformer, utilisent souvent des couches attentionnelles ou des couches de pointage, qui permettent de réduire le nombre de paramètres tout en améliorant les performances de prédiction.[w9][w10]

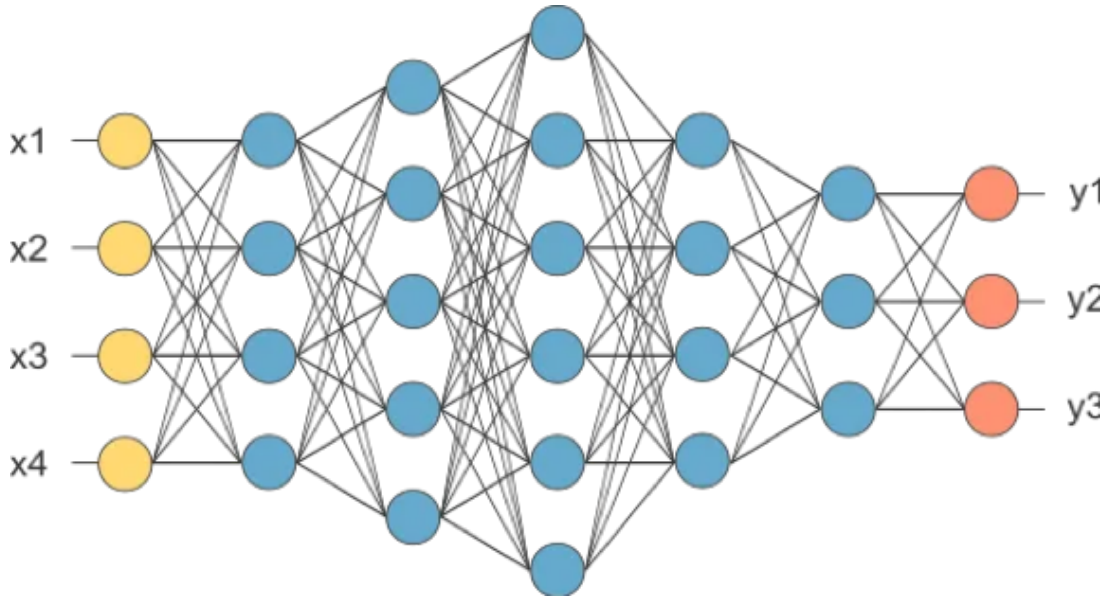


FIGURE 2.8 – Illustration de couche fully-connected

6.4 La couche de correction 'ReLU'

La couche de correction ReLU (Rectified Linear Unit) est une fonction d'activation qui applique une transformation linéaire à chaque entrée de neurone, en remplaçant toutes les valeurs négatives par zéro et en laissant les valeurs positives inchangées. La fonction mathématique de la couche ReLU est définie comme suit : f(x) = max(0,x). [w11]

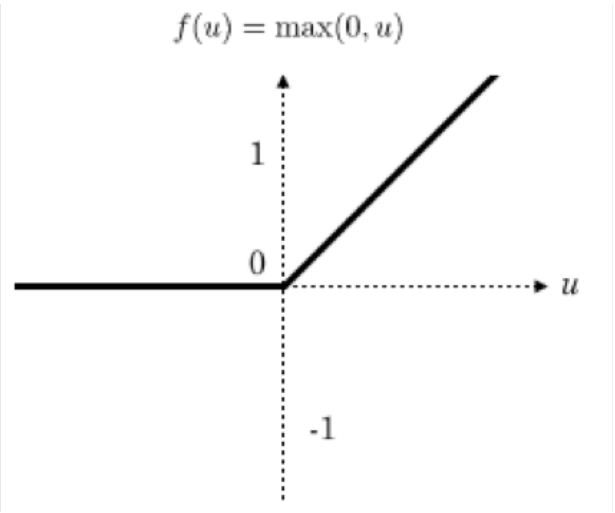


FIGURE 2.9 – Illustration de couche ReLu

7 Les hyperparamètres d'un modèle

les hyperparamètre sont des paramètres dont la valeur est définie avant le début du processus d'apprentissage. Pour les réseaux de neurones profonds, l'on peut distinguer 03 types d'hyperparamètres :

- Les hyperparamètres des couches : taille du noyau, dropout, méthode d'activation des couches cachés, méthode d'activation du couche finale, ...

- Les hyperparamètres de compilation du modèle : optimizer, loss, learning rate ...

- Les hyperparamètres d'exécution du modèle : batch size, nombre d'epochs, ... [w12]

7.1 Optimiseur (optimizer)

Les optimiseurs sont des algorithmes utilisés pour modifier les attributs du réseau de neurones tels que les poids et le taux d'apprentissage afin de réduire les pertes. Les optimiseurs les plus couramment utilisés en apprentissage profond incluent : [w13]

La descente de gradient stochastique (SGD)

Il s'agit de l'optimiseur le plus simple et le plus couramment utilisé. Il ajuste les poids du modèle en utilisant la direction opposée du gradient de la fonction de coût par rapport aux paramètres. La version stochastique calcule le gradient sur un échantillon aléatoire des données d'entraînement à chaque étape.

L'optimiseur Adam

Il combine la descente de gradient stochastique avec une estimation adaptative des moments du gradient pour ajuster les taux d'apprentissage pour chaque paramètre du modèle.

L'optimiseur RMSprop

Il utilise une estimation adaptative des moments du gradient en utilisant une moyenne mobile exponentielle des carrés des gradients pour ajuster les taux d'apprentissage pour chaque paramètre.

L'optimiseur Adagrad

Il adapte les taux d'apprentissage pour chaque paramètre en fonction de la fréquence d'apparition de chaque paramètre dans les données d'entraînement.

Il existe de nombreux autres optimiseurs disponibles et le choix de l'optimiseur dépend de divers facteurs tels que la taille des données d'entraînement, la complexité du modèle et le temps disponible pour l'entraînement.

7.2 Régularisation

La régularisation est une technique utilisée pour éviter le surapprentissage (overfitting) et améliorer les performances de généralisation d'un modèle. L'objectif de la régularisation est de réduire la complexité du modèle et d'empêcher les poids des paramètres du modèle de devenir trop grands. Il existe plusieurs techniques de régularisation telles que la régularisation de dropout, qui consiste à supprimer aléatoirement des neurones du modèle pendant l'entraînement pour réduire la coadaptation des neurones.[w14]

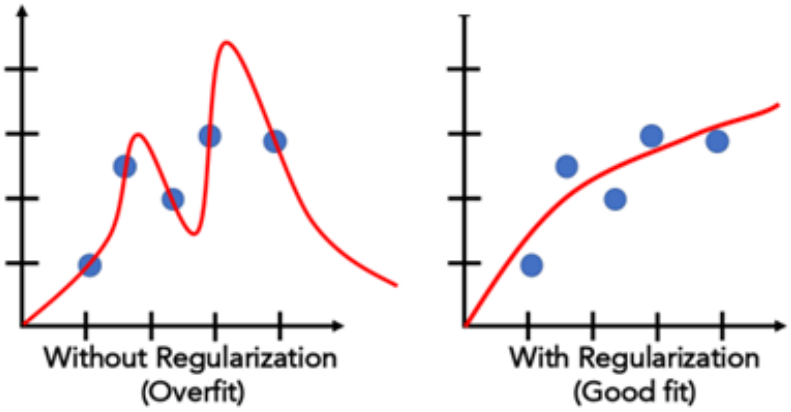


FIGURE 2.10 – Régularisation sur un modèle sur-ajusté

7.3 Epoque

Une époque (ou "epoch" en anglais) est une itération complète de l'algorithme d'apprentissage sur l'ensemble des données d'entraînement. Une époque se produit lorsque l'algorithme d'apprentissage a effectué des calculs sur toutes les données d'entraînement une fois. Le nombre d'époque peut atteindre plusieurs milliers, car la procédure se répète

indéfiniment jusqu'à ce que le taux d'erreurs du modèle soit suffisamment réduit. En général, plus le nombre d'époques est élevé, plus l'algorithme d'apprentissage a de chances de converger vers un modèle qui généralise bien et peut être utilisé pour faire des prédictions précises sur de nouvelles données. [w15]

7.4 Batch size

Les données d'entraînement sont décomposées en plusieurs petits lots (ou "batches " en anglais). Le but est d'éviter les problèmes liés à un manque d'espace de stockage. Les batches peuvent être facilement utilisés pour nourrir le modèle de Machine Learning afin de l'entraîner. Ce processus de décomposition des données est appelé « batch ».[w15]

8 Fonctions d'activation

Les fonctions d'activation les plus couramment utilisées sont : [w16]

8.1 Fonction 'ReLU'

La fonction ReLU prend une valeur d'entrée x et retourne soit cette valeur si elle est positive, soit zéro si elle est négative. Ainsi, la fonction ReLU est linéaire pour les valeurs positives et non linéaire pour les valeurs négatives. La fonction ReLU est souvent utilisée comme fonction d'activation dans les réseaux de neurones car elle est simple et efficace. Elle est également plus rapide à calculer que d'autres fonctions d'activation non linéaires telles que la fonction sigmoïde ou la fonction tangente hyperbolique.

$$f(x) = \max(0, x) \tag{2.1}$$

8.2 Fonction 'sigmoid'

Elle transforme une valeur d'entrée en une valeur entre 0 et 1. Elle est souvent utilisée dans les réseaux de neurones à une seule couche cachée.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

8.3 La tangente hyperbolique

La tangente hyperbolique (ou fonction tanh) est une fonction mathématique, où x est l'entrée à la fonction et tanh(x) est la sortie. La fonction tanh est similaire à la fonction sigmoïde, mais elle est centrée autour de zéro et a une plage de sortie plus large, allant de -1 à 1. Elle est donc mieux adaptée pour les tâches où les données ont une moyenne nulle et une variance élevée. Elle est définie comme suit :

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

8.4 Fonction 'Softmax'

Elle prend en entrée un vecteur de nombres réels et retourne un vecteur de probabilités normalisées, dont la somme est égale à 1. Chaque élément du vecteur de sortie représente la probabilité que l'entrée appartienne à une classe particulière.Mathématiquement, la fonction softmax est définie comme suit :

$$[\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}] \tag{2.4}$$

9 Transformateur de Vision

9.1 Histoire des transformateurs

Les transformateurs sont un type d'architecture d'apprentissage profond pour les tâches de traitement du langage naturel (NLP) qui ont été introduit en 2017 par Vaswani et al. Dans un article intitulé "Attention is All You Need" [21]. Ils reposent sur des mécanismes d'auto-attention, qui permettent au modèle de peser l'importance des différents éléments dans une séquence lors de la prédiction.

Avant les transformateurs, les modèles NLP les plus couramment utilisés étaient les réseaux neuronaux récurrents (RNN) et les réseaux neuronaux convolutionnels (CNN). Bien que les RNN soient bons pour gérer les séquences, ils peuvent avoir des difficultés avec des séquences très longues et peuvent être coûteux en termes de calcul. D'autre part, les CNN sont efficaces mais peuvent avoir du mal à capturer les dépendances à longue portée dans les séquences. L'idée clé derrière les transformateurs est d'utiliser l'auto-attention pour peser dynamiquement la contribution de chaque élément dans une séquence à la prédiction. Cela permet au modèle de gérer des séquences de n'importe quelle longueur et de traiter efficacement les informations de toutes les parties de la séquence, ce qui entraîne une amélioration des performances sur une variété de tâches NLP. Depuis son introduction, les transformateurs ont été largement adoptés dans la communauté NLP et ont obtenu des résultats à l'état de l'art sur de nombreux ensembles de données de référence. Ils ont également été utilisés dans divers autres domaines, tels que la vision par ordinateur et l'apprentissage par renforcement, et ont été affinés sur un grand corpus de textes pour générer du texte, résumer du texte,...etc.

9.2 Qu'est ce qu'un transformateur ?

Un transformateur est un modèle d'apprentissage profond qui appartient au domaine du machine learning (apprentissage automatique) et plus précisément du deep learning. Taillé pour le traitement automatique des langues (natural language processing NLP), le transformateur est un réseau de neurones profonds conçu pour ingérer des données d'apprentissage séquentielles en utilisant des mécanismes d'attention.Le transformer permet à la machine d'apprendre des séquences informatiques de manière automatique, sans avoir

été programmés spécifiquement à cet effet.[21]

9.3 Transformateur de vision

Le Transformateur de vision est une architecture de réseau de neurones utilisée pour résoudre des tâches de traitement d'images, telles que la classification d'images, la segmentation sémantique et la détection d'objets. C'est une extension de l'architecture Transformateur utilisée pour le traitement de séquences dans le domaine du traitement du langage naturel (NLP). Le Transformateur de vision utilise des blocs de Transformateur qui intègrent des mécanismes d'attention pour capturer les relations entre les différentes parties de l'image. Il s'appuie également sur des techniques de pré-entraînement non supervisé pour extraire des caractéristiques visuelles à partir des images, qui peuvent être utilisées pour résoudre une variété de tâches de vision par ordinateur. En comparaison avec les architectures de réseau de neurones traditionnelles, le Transformateur de vision a montré des performances supérieures sur plusieurs tâches de vision par ordinateur, tout en nécessitant moins de données d'entraînement. Il est donc devenu l'une des architectures les plus populaires pour la résolution de tâches de traitement d'images.[22][23]

9.4 Les types de transformateurs

Il existe différents types de transformateurs, notamment :
- Transformateurs d'encodeur uniquement
- Transformateurs à décodeur uniquement
- Transformateurs d'encodeur-décodeur
- Transformateurs de vision

Transformateurs d'encodeur uniquement

Ces modèles n'ont que le composant d'encodeur, qui est responsable de la transformation des données d'entrée en une représentation d'espace latent. Ils sont couramment utilisés dans des tâches telles que la modélisation du langage, où le but est de prédire le mot suivant dans une séquence.

Transformateurs à décodeur uniquement

Ces modèles n'ont que le composant décodeur, qui est chargé de générer des données de sortie en fonction d'une entrée donnée et d'une représentation apprise. Ils sont couramment utilisés dans des tâches telles que la traduction de la langue, où le but est de générer une phrase dans la langue cible à partir d'une phrase dans la langue source.

Transformateurs d'encodeur-décodeur

Ces modèles ont à la fois un composant d'encodeur et un composant de décodeur, et sont couramment utilisés dans les tâches de séquence à séquence telles que la traduction automatique, la synthèse de texte et les agents conversationnels.

Transformateurs de vision (ViT)

Il s'agit d'une variante de l'architecture des transformateurs qui a été adaptée aux tâches de vision par ordinateur, telles que la classification d'images, la détection d'objets et la segmentation d'images.

10 Mecanisme de l'auto-attention

L'auto-attention est un mécanisme d'apprentissage profond qui permet à un modèle de se concentrer sur différentes parties de sa propre entrée pour peser leur importance relative. Elle est devenue populaire car elle est puissante pour diverses tâches de traitement du langage naturel, telles que la traduction automatique et la classification de texte, ainsi que pour des tâches de vision par ordinateur telles que la détection d'objets et la segmentation. L'auto-attention peut capturer des dépendances à longue portée et des relations entre différents éléments de l'entrée, ce qui en fait un outil utile pour modéliser des données complexes et structurées.[21]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.5}$$

Les principales étapes du calcule de l'auto attention sont :

- **Etape 1 :** le calcul de l'auto-attention consiste à créer trois vecteurs à partir de cha-

cun des vecteurs d'entrée de l'encodeur. Ainsi, pour chaque mot, nous créons un vecteur 'Query', un vecteur 'Key' et un vecteur 'Value'. Ces vecteurs sont créés en multipliant l'intégration par trois matrices que nous avons entraînées au cours du processus d'entraînement. La multiplication de x1 par la matrice de poids 'WQ' produit 'q1', le vecteur " query" associé à ce mot. Nous finissons par créer une " query", une "key" et une projection de "value" de chaque mot dans la phrase d'entrée. Cela donne à la couche d'auto-attention des paramètres contrôlables et lui permet de modifier les vecteurs entrants.

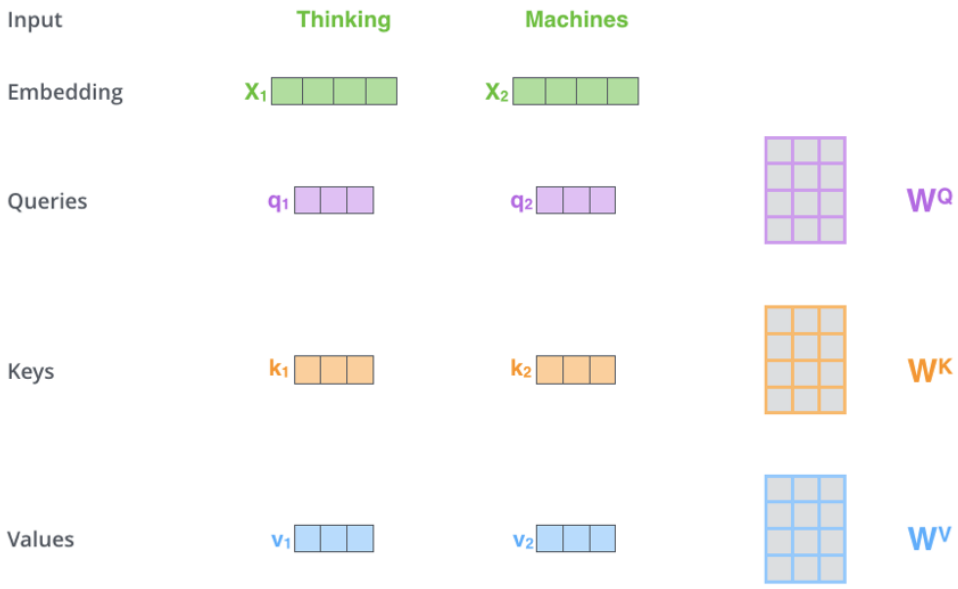


FIGURE 2.11 – Illustration pour la première étape du calcule de l'auto attention

- **Etape 2 :** Cette étape consiste à calculer un score. Nous calculons l'auto-attention pour le premier mot de cet exemple, " Thinking". Nous devons marquer chaque mot de la phrase d'entrée par rapport à ce mot. Le score détermine la concentration à accorder aux autres parties de la phrase d'entrée lorsque nous encodons un mot à une certaine position. Le score est calculé en prenant le produit scalaire du vecteur de requête avec le vecteur clé du mot respectif que nous notons. Donc, si nous traitons l'auto-attention pour le mot en position n°1, le premier score serait le produit scalaire de q1 et k1. Le deuxième score serait le produit scalaire de q1 et k2.
- **Etapes 3 et 4 :** Elles consistent à diviser les scores par 8 (la racine carrée de la dimension des vecteurs clés), puis transmettez le résultat via une opération softmax. Softmax normalise les scores pour qu'ils soient tous positifs et totalisent 1. Le score softmax détermine combien chaque mot sera exprimé à cette position. De toute évidence,

le mot à cette position aura le score softmax le plus élevé, mais il est parfois utile de s'occuper d'un autre mot pertinent pour le mot actuel.

- **Etape 5 :** Cette étape consiste à multiplier chaque vecteur de valeur par le score softmax. L'intuition ici est de garder intactes les valeurs du ou des mots sur lesquels on veut se concentrer.

- **Etape 6 :** Elle consiste à sommer les vecteurs de valeur pondérés. Cela produit la sortie de la couche d'auto-attention à cette position (pour le premier mot).

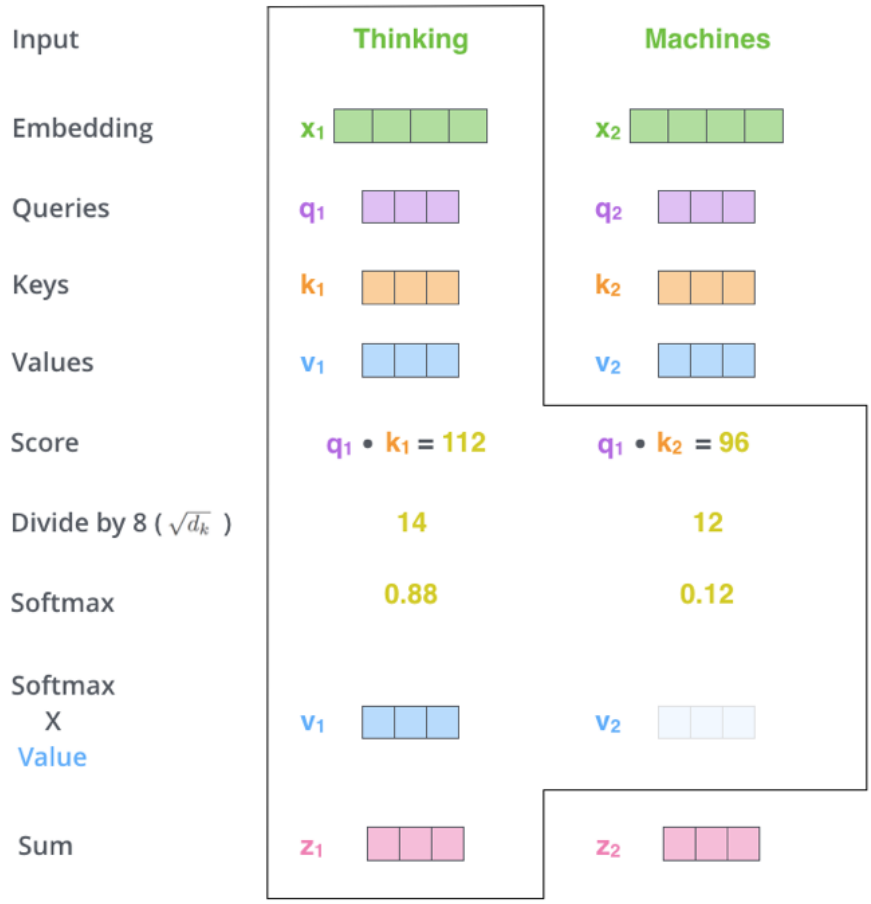


FIGURE 2.12 – Illustration pour les étape du calcule de l'auto attention

Le vecteur résultant est celui que nous pouvons transmettre au réseau neuronal "feed-forward". Ce calcul est effectué sous forme matricielle pour un traitement plus rapide. Maintenant, nous allons expliquer le calcul matriciel de l'auto-attention.

- Principalement il consiste à calculer les matrices 'Query', 'key' et 'value'. En regroupant nos intégrations 'embeddings' dans une matrice X et en la multipliant par les matrices de pondération que nous avons entraînées (WQ, WK, WV).

-Enfin, puisque nous avons affaire à des matrices, nous pouvons condenser les étapes deux

à six dans une seule formule pour calculer les sorties de la couche d'auto-attention.

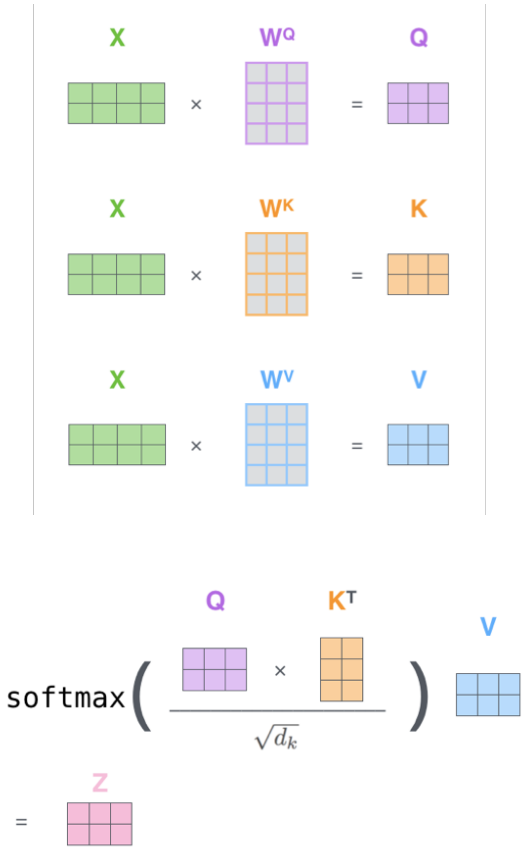


FIGURE 2.13 – Le calcul de l'auto-attention sous forme matricielle

11 La Différence entre le transformateur et transformateur Visual

La principale différence entre les architectures de transformateur et transformateur Visual est le type de données qu'elles traitent. Les transformateurs sont principalement utilisés pour le traitement du langage naturel (NLP), tandis que les transformateurs Visual sont spécifiquement conçus pour traiter les images.

Le transformateur Visual utilise une architecture d'encodeur uniquement, contrairement à l'architecture de base du Transformateur qui utilise une architecture d'encodeur-décodeur. Dans l'architecture de base du Transformateur, l'encodeur traite la séquence d'entrée et produit une représentation intermédiaire, qui est ensuite utilisée par le décodeur pour générer une séquence de sortie. Cependant, dans le transformateur Visual, il n'y a pas de séquence de sortie à générer. Au lieu de cela, l'objectif est de produire des représentations

de haut niveau pour chaque patch de l'image en utilisant l'encodeur uniquement. Ces représentations de haut niveau peuvent ensuite être utilisées pour différentes tâches de vision par ordinateur, telles que la classification d'image ou la segmentation d'image. En résumé, le transformateur Visual n'utilise pas de décodeur car elle ne produit pas de séquence de sortie. Elle utilise uniquement une architecture d'encodeur pour produire des représentations de haut niveau pour chaque patch de l'image.

12 Structure globale d'un transformateur Visual

Le transformateur Visual est une architecture de réseau de neurones qui utilise des couches d'attention multi-têtes pour permettre aux différents patchs d'interagir les uns avec les autres. Cette architecture a été utilisée avec succès pour la reconnaissance d'images et d'autres tâches de vision par ordinateur.[22][23]

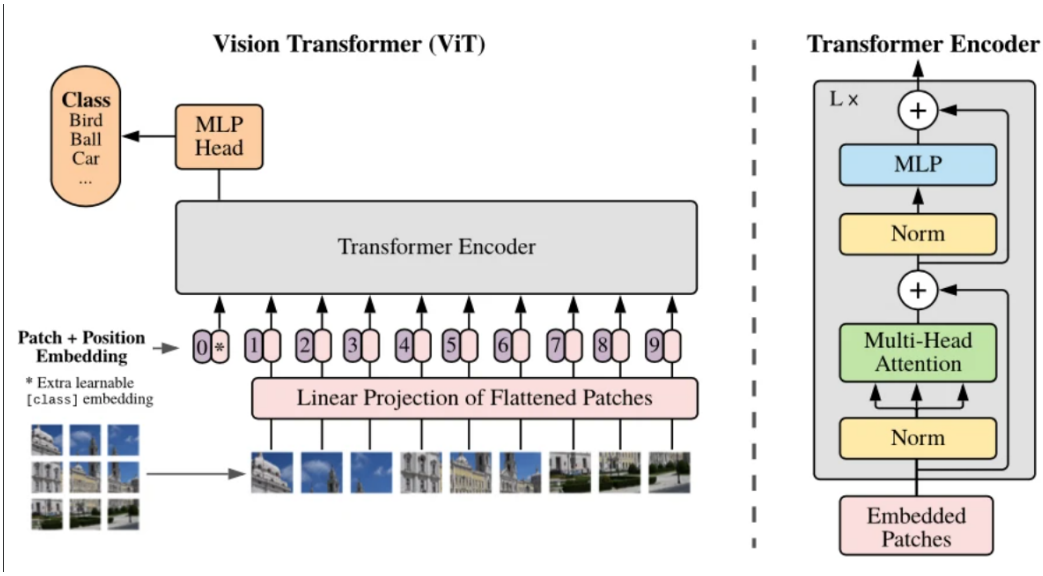


FIGURE 2.14 – Architecture de transformateur visuel

12.1 Extraction de patchs

La conversion d'une image en patchs est une étape clé dans l'architecture du transformateur Visual. Elle permet de découper l'image en régions rectangulaires, appelées patchs, qui sont ensuite traitées individuellement par le réseau de neurones.

Pour convertir une image en patchs, on procède généralement comme suit :

Définir la taille des patchs : On choisit une taille fixe pour les patchs, par exemple 16x16 pixels ou 32x32 pixels.

Découper l'image en patchs : On divise l'image en une grille de patchs rectangulaires de taille fixe. Chaque patch est découpé à partir de l'image en prenant une région rectangulaire de la taille choisie.

Recouvrement des patchs : Pour assurer que toutes les zones de l'image sont couvertes, on peut appliquer un recouvrement entre les patchs. Cela signifie que chaque patch se chevauche partiellement avec les patchs adjacents, de sorte que chaque pixel de l'image est couvert par plusieurs patchs. Cette technique est particulièrement utile pour les bords de l'image.

Redimensionner les patchs : Les patchs sont généralement redimensionnés pour avoir la même taille. Cette étape permet de s'assurer que tous les patchs ont la même dimension et que les calculs du réseau de neurones peuvent être effectués de manière efficace.

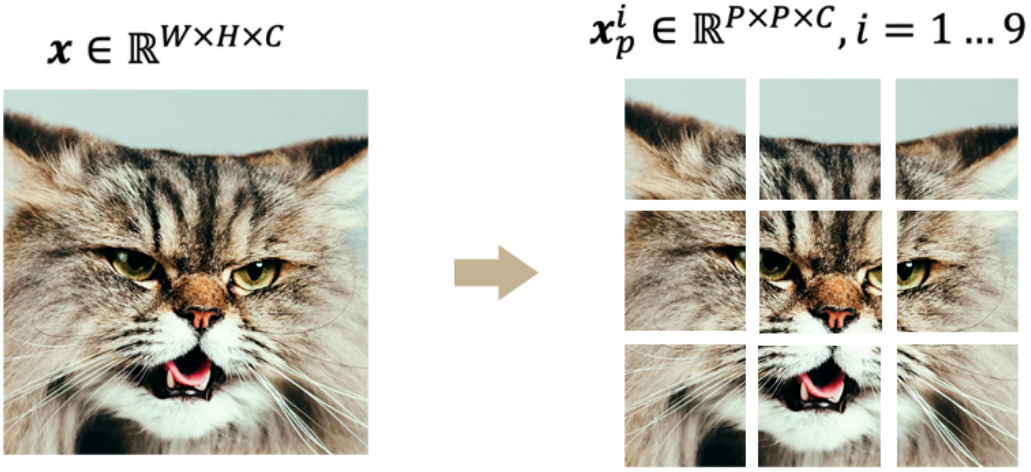


FIGURE 2.15 – La conversion d'une image en patchs

12.2 La projection linéaire de patches aplatis (Linear projection of flattened patches)

La projection linéaire des patchs aplatis consiste à transformer un patch d'image bidimensionnel en un vecteur unidimensionnel. Aplatir le patch signifie convertir le patch 2D en un vecteur 1D en concaténant les valeurs de pixel le long d'une seule dimension. Une projection linéaire du patch aplati implique de multiplier ce vecteur 1D par une matrice de poids, suivie de l'ajout d'un vecteur de biais. La matrice de poids définit une

transformation linéaire qui mappe le vecteur d'entrée vers un vecteur de sortie de dimension différente, tandis que le vecteur de biais est ajouté pour décaler la sortie de la transformation. Le vecteur de sortie résultant peut être interprété comme un vecteur de caractéristiques qui représente le contenu du patch d'image.

12.3 Encodage de position 'Position embedding'

L'encodage de position fonctionne en ajoutant un vecteur de longueur fixe aux 'embeddings' de jetons d'entrée pour représenter leur position dans la séquence. Ce vecteur est ajouté aux 'embeddings' d'entrée et ensuite alimenté dans la couche d'encodeur du transformateur. Le vecteur de l'encodage de position est appris pendant l'entraînement, mais ses valeurs ne changent pas pendant l'exécution. Le vecteur de l'encodage de position peut prendre différentes formes, mais une approche courante est d'utiliser un ensemble fixe de fonctions sinusoidales avec différentes fréquences pour coder les informations de position. Les fonctions sinusoidales permettent au modèle de distinguer entre différentes positions dans la séquence sans avoir besoin de paramètres appris supplémentaires.

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{2.6}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{2.7}$$

12.4 Intégration apprenable 'learnable embedding'

Le jeton de classification est généralement ajouté en tant que premier jeton de la séquence d'entrée, suivi de l'ensemble des autres jetons de la séquence. Le jeton de classification est un jeton spécial qui indique au modèle que la tâche à accomplir est une tâche de classification. Deux choses rendent le jeton de classification spécial. Premièrement, il ne représente pas un jeton réel, ce qui signifie qu'il commence comme une "ardoise vierge" pour chaque patch. Deuxièmement, la sortie finale du jeton de classification est utilisée comme entrée dans une tête de classification pendant la préformation.

FIGURE 2.16 – Exemple de l'encodage de position

12.5 Encodeur

L'encodeur est une partie importante de l'architecture du transformateur.Il reçoit une séquence d'entrée de jetons et la transforme en une représentation vectorielle qui peut être utilisée pour d'autres étapes du modèle. L'encodeur est composé de plusieurs blocs, chaque bloc d'encodeur est composé des éléments suivants : Couche d'attention multi-têtes, Couche de normalisation et Couche de perceptrons multicouches 'MLP' (feedforward). Chaque bloc d'encodeur reçoit la représentation vectorielle de sortie du bloc précédent en entrée et génère une nouvelle représentation vectorielle en sortie. Les blocs d'encodeur sont empilés les uns sur les autres pour former un encodeur à plusieurs couches.

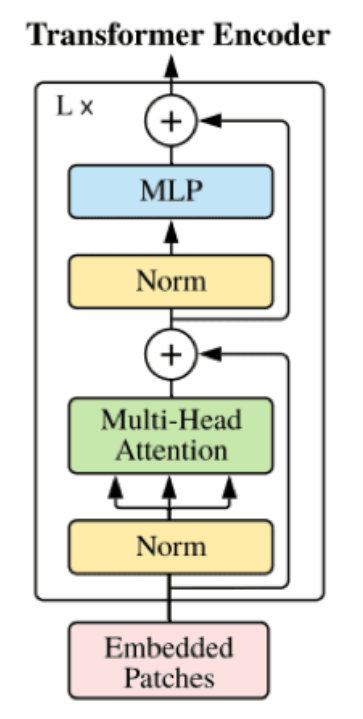


FIGURE 2.17 – L'architecture de l'encodeur du transformateur

12.6 L'attention multi-têtes (MHA)

L'attention multi-têtes permet au modèle de prendre en compte les relations entre tous les éléments de la séquence d'entrée en calculant des poids d'attention pour chacun des éléments. La MHA calcule des poids d'attention pour chaque paire d'éléments dans la séquence d'entrée en utilisant une fonction de similarité. Les poids d'attention sont ensuite normalisés pour que leur somme soit égale à 1. Enfin, chaque élément de la séquence d'entrée est multiplié par son poids d'attention correspondant, puis tous les éléments pondérés sont additionnés pour former une représentation de la séquence d'entrée. La MHA divise la représentation de chaque élément en plusieurs têtes ou canaux, chacun calculant une attention différente. Chaque tête calcule donc ses propres poids d'attention, qui sont ensuite concaténés pour former une représentation vectorielle finale pour chaque élément de la séquence.

$$\mathrm{MHA}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \mathrm{head}_2, \ldots, \mathrm{head}_h) \cdot W^O \tag{2.8}$$

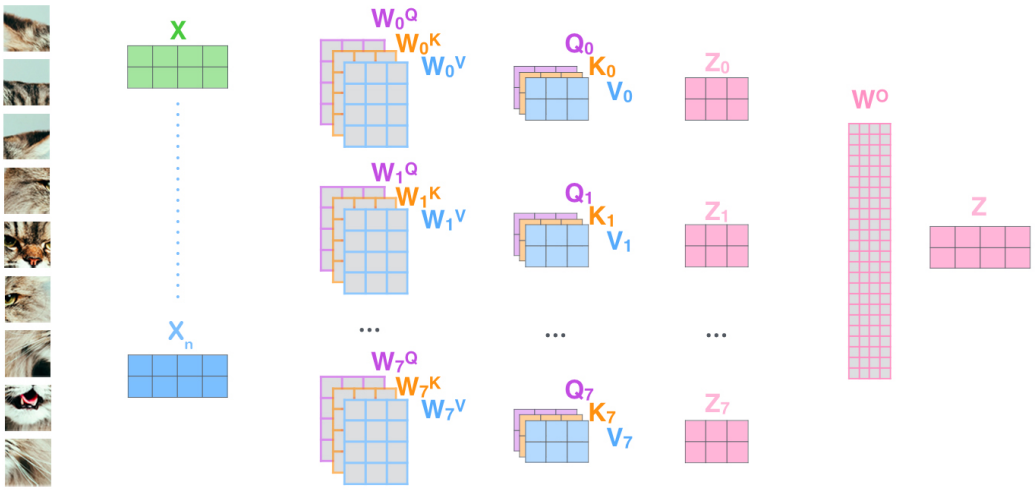


FIGURE 2.18 – Illustration pour le fonctionnement de MHA

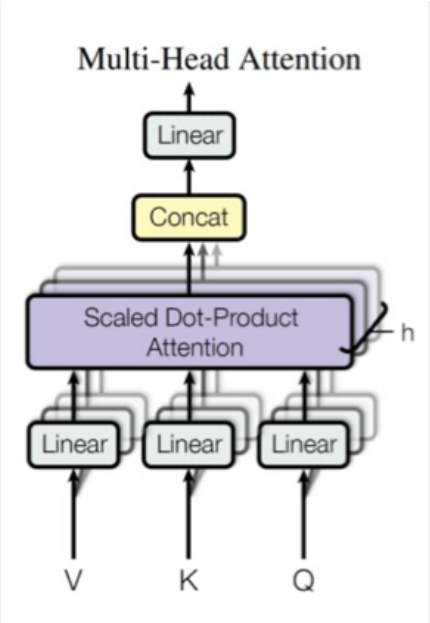


FIGURE 2.19 – Architecture du MHA

12.7 Attention du produit scalaire mise à l'échelle (Scaled Dot-Product Attention)

L'attention du produit scalaire mise à l'échelle est un type de mécanisme d'attention qui permet à un modèle de se concentrer sélectivement sur les parties pertinentes d'une séquence d'entrée lors de la prise de décisions. Dans ce mécanisme d'attention, la séquence d'entrée est transformée en un ensemble de requêtes (queries), de clés (keys) et de valeurs (values), qui sont ensuite utilisés pour calculer un score d'attention entre chaque requête et chaque clé. Les scores d'attention sont ensuite normalisés à l'aide d'un facteur d'échelle pour éviter qu'ils ne deviennent trop grands, ce qui peut causer une instabilité numérique pendant l'entraînement. Une fois que les scores d'attention sont normalisés, ils sont utilisés pour pondérer les valeurs correspondantes et produire une somme pondérée, qui représente la sortie finale du mécanisme d'attention. Cette sortie est ensuite combinée avec la séquence d'entrée d'origine pour produire la sortie finale du modèle.

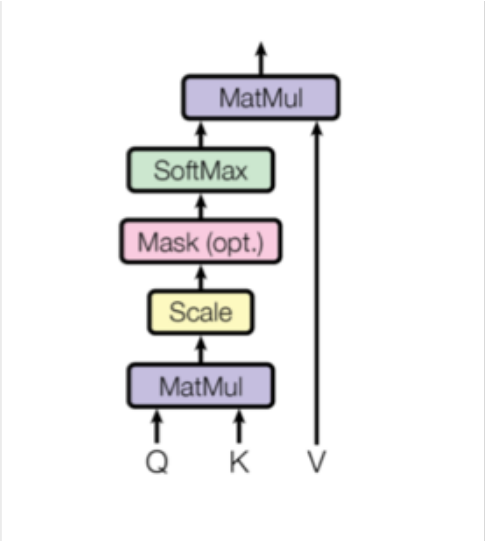


FIGURE 2.20 – L'architecture de Attention du produit scalaire mise à l'échelle

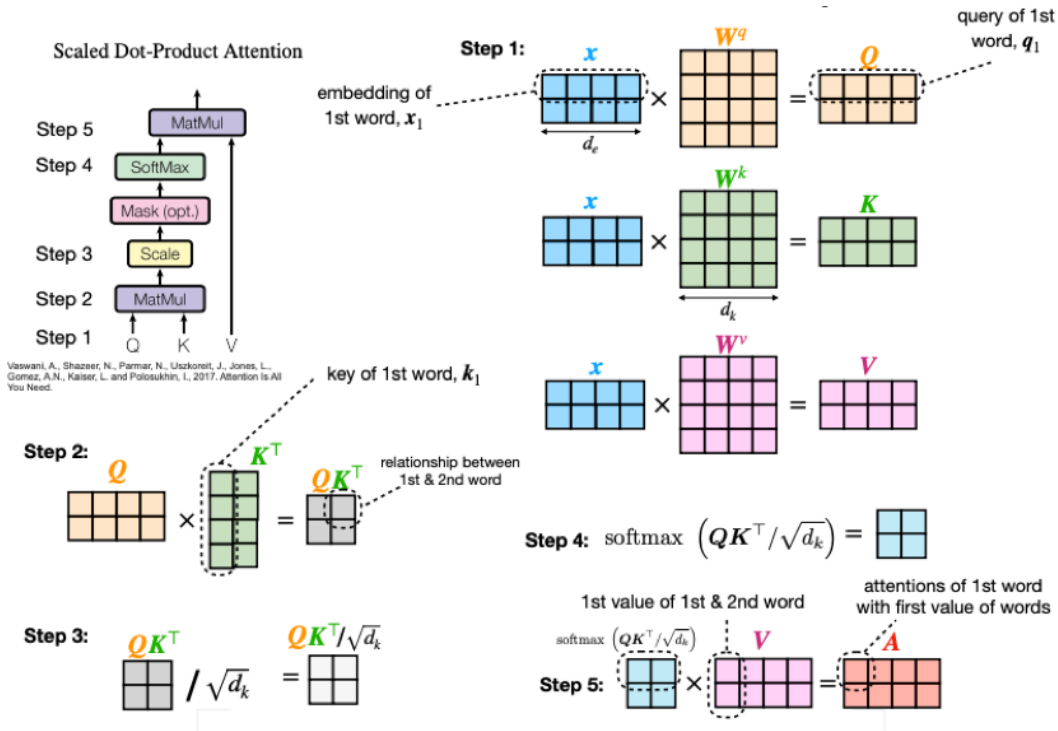


FIGURE 2.21 – Illustration du fonctionnement de 'Scaled Dot-Product Attention'

12.8 Couche de normalisation et connexion résiduelle (Add and Norm layer)

La couche 'Add and Norm' est une implémentation spécifique de la combinaison des techniques de connexion résiduelle et de normalisation de couche utilisées dans l'encodeur de

Transformateur. La connexion résiduelle est utilisée en conjonction avec la couche de normalisation. Plus précisément, à chaque étape de l'encodeur ou du décodeur, la connexion résiduelle permet de transmettre l'information directement de la couche précédente à la couche suivante. Cette connexion est ensuite suivie d'une couche de normalisation pour normaliser les activations et stabiliser l'apprentissage. Cette combinaison permet à l'information de passer plus facilement à travers le réseau, ce qui améliore la stabilité et la performance.

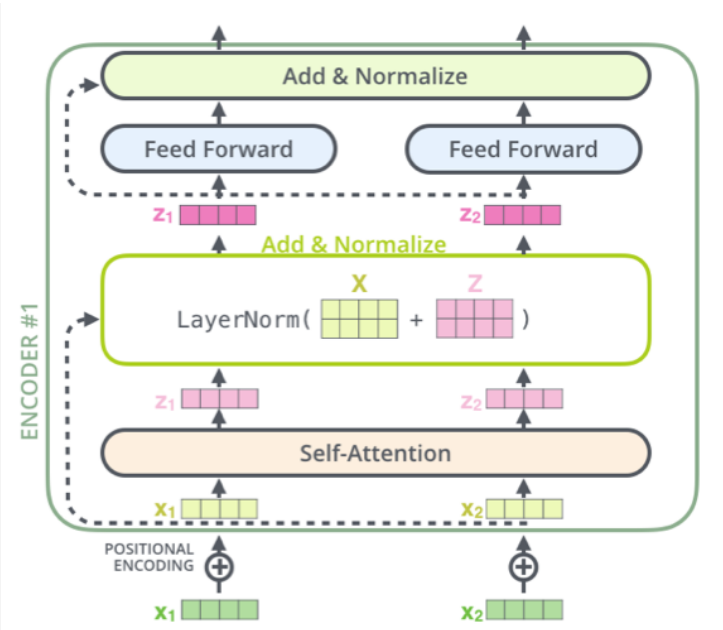


FIGURE 2.22 – Illustration de Couche de normalisation et la connexion résiduelle

12.9 Couche de perceptrons multicouches (MLP)

La couche de perceptrons multicouches (Multi-Layer Perceptron en anglais, abrégé MLP) est connue sous le nom de couche entièrement connectée (feedforward) ou couche dense. Comme son nom l'indique, la couche MLP est composée de plusieurs couches de perceptrons, chacune d'entre elles étant entièrement connectée à la couche précédente. Chaque perceptron dans une couche MLP effectue une combinaison linéaire de ses entrées, suivie d'une fonction d'activation non-linéaire appelée GELU (Gaussian Error Linear Unit), pour produire une sortie. La sortie de chaque perceptron dans une couche est transmise à tous les perceptrons de la couche suivante. Le MLP est appelé "multi-couche" car il a plusieurs couches de perceptrons entre la couche d'entrée et la couche de sortie. Les couches cachées du MLP permettent au modèle d'apprendre des relations non linéaires complexes

entre les entrées et les sorties. Le nombre de couches cachées et le nombre de neurones dans chaque couche peuvent être ajustés pour équilibrer la complexité du modèle et sa capacité à généraliser à de nouvelles données.

$$\text{GELU}(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \cdot x \tag{2.9}$$

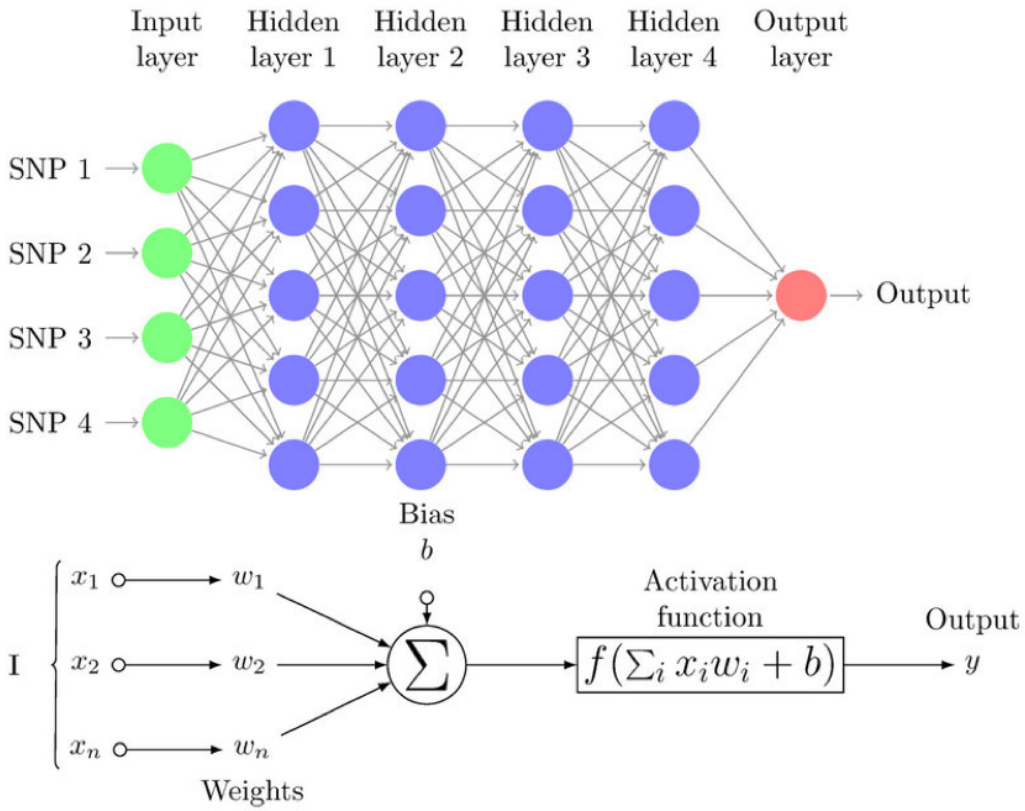


FIGURE 2.23 – Architecture du perceptrons multicouches (MLP)

12.10 Tête MLP (MLP Head)

La tête MLP est La couche finale du MLP, également appelée couche de sortie, est responsable de produire les prédictions finales d'un réseau neuronal. La couche de sortie se compose généralement d'un ensemble de neurones qui sont entièrement connectés à la couche précédente. La sortie de la couche finale du MLP dépend de la tâche spécifique que le réseau neuronal est conçu pour résoudre. Par exemple, dans une tâche de classification, la couche de sortie peut utiliser une fonction d'activation softmax pour produire une

distribution de probabilités sur les classes possibles. Dans une tâche de détection d'objets, la couche de sortie peut produire un ensemble de coordonnées de boîtes englobantes et de probabilités de classe pour chaque objet détecté dans l'image d'entrée.

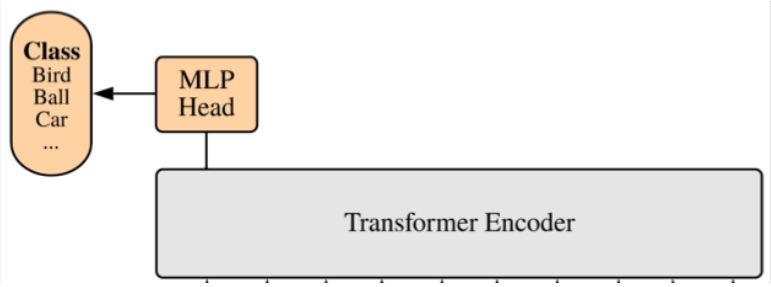


FIGURE 2.24 – La Tête MLP

13 Structure d'un transformateur

Dans l'architecture standard de Transformateur, les données d'entrée sont une séquence de jetons, tels que des mots dans le cas du NLP. Les données d'entrée sont traitées comme dans les transformateurs visuels car elles partagent une structure similaire. La structure du codeur est la même, la seule différence est la séquence de sortie. Dans l'architecture de base du Transformateur, l'encodeur traite la séquence d'entrée et produit une représentation intermédiaire, qui est ensuite utilisée par le décodeur pour générer une séquence de sortie.[21]

13.1 Décodeur

Le décodeur est l'une des deux parties principales de l'architecture de Transformateur. Le décodeur se compose de plusieurs blocs qui sont similaires à celles de l'encodeur, mais qui ont des différences importantes. Chaque bloc de décodeur est composé des éléments suivants : Couche d'attention multi-têtes, Couche d'attention multi-têtes masquée, Couche 'feedforward'. Comme pour l'encodeur, chaque couche du décodeur est suivie d'une couche de normalisation 'Add Norm' pour ajouter la sortie de la sous-couche précédente et normaliser la sortie combinée. Le décodeur permet de générer des séquences de sortie en utilisant les informations contenues dans la représentation vectorielle de l'entrée et les sorties précédentes du décodeur. [21]

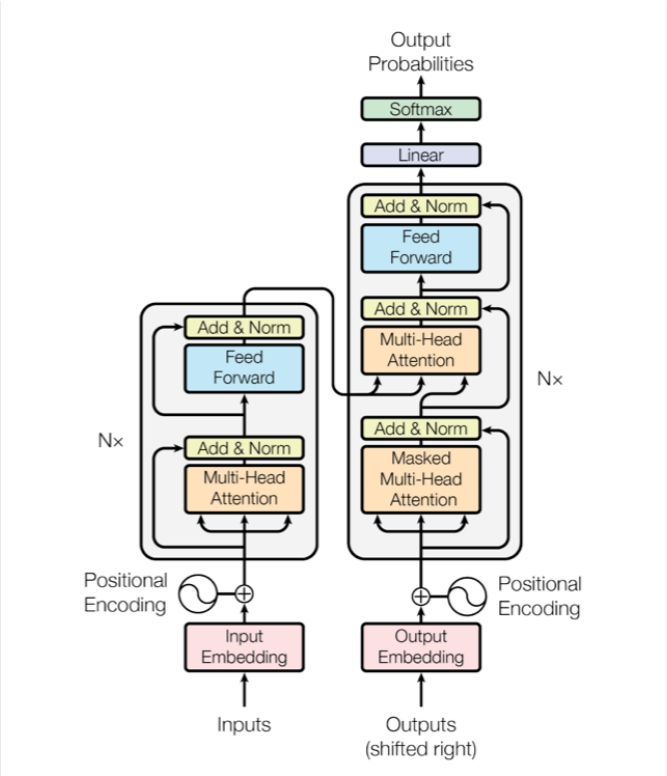


FIGURE 2.25 – L'architecture de transformateur avec décodeur

13.2 La couche linéaire et Softmax

La couche linéaire est une couche entièrement connectée qui prend en entrée les sorties de la couche précédente et calcule une combinaison linéaire à l'aide d'un ensemble de poids et de biais appris. Cela produit un vecteur de scores pour chaque classe dans le problème de classification. La couche softmax prend ensuite le vecteur de scores et lui applique la fonction softmax, qui convertit les scores en probabilités qui s'additionnent à 1. Chaque élément du vecteur de sortie représente la probabilité de l'entrée appartenant à une classe particulière. La sortie finale du modèle est l'étiquette de classe prédite, qui est la classe avec la probabilité la plus élevée.

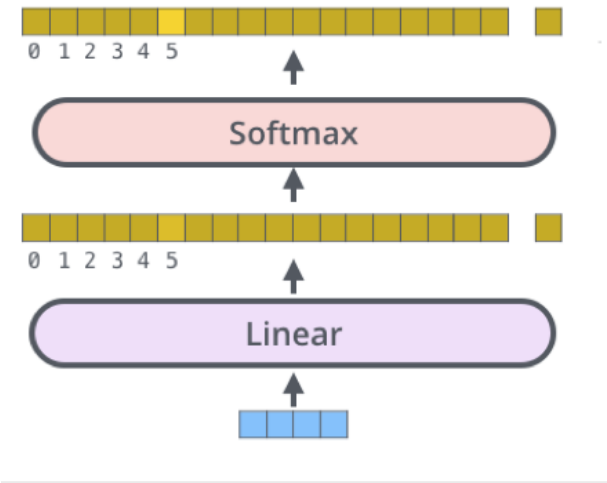


FIGURE 2.26 – Illustration de La couche linéaire et Softmax

14 Applications des transformateurs de vision

Voici quelques exemples d'applications des transformateurs de vision :

- Classification des images
- Légende des images
- Segmentation d'image
- Détection d'objets
- Conduite autonome
- Reconnaissance des actions

15 Limites du transformateur

Les transformateurs ont deux limites principales :[24]

15.1 La complexité quadratique

Les transformateurs sont des modèles puissants, mais ils s'accompagnent de coûts de calcul important. La complexité quadratique des transformateurs (O(T2)) est particulièrement préoccupante lorsque l'on travaille avec des données vidéo. Cependant, certains chercheurs ont tenté de résoudre ce problème en adaptant l'architecture du transformateur pour mieux gérer la vidéo. Ceci est généralement réalisé en réduisant la longueur de la séquence lors du calcul de la matrice d'attention. Une approche consiste à restreindre le nombre de jetons utilisés dans une seule opération d'attention, tandis qu'une autre consiste à agréger progressivement les informations via des opérations contextuelles. Ces adaptations contribuent à rendre les transformateurs plus utilisables pour les tâches vidéo en réduisant la complexité de calcul tout en conservant leur capacité à modéliser des relations complexes dans les données.

15.2 Besoin de grands volumes de données

Le transformateur nécessite des quantités massives de données pour être entraîné efficacement. Dans le domaine de la vision par ordinateur, la disponibilité de telles données peut être limitée, ce qui peut rendre le processus d'apprentissage plus difficile.

16 Différence entre CNN et ViT

Transformateur Visual peut obtenir une précision similaire avec moins de ressources de calcul en utilisant une stratégie d'apprentissage différente de celle des réseaux de neurones convolutionnels traditionnels. Le modèle de transformateur visual a tendance à montrer un biais inductif plus faible que les CNN, ce qui les rend moins efficaces pour apprendre à partir de petits ensembles de données sans régularisation ni augmentation des données. Le modèle ViT représente une image d'entrée sous la forme d'une série de patchs d'image et utilise cette représentation directement pour prédire les étiquettes de classe pour cette image. ViT fonctionne remarquablement bien lorsqu'il est formé sur de grandes quantités de données, il peut surpasser CNN avec seulement une fraction des ressources de calcul. CNN utilise des tableaux de pixels, tandis que ViT divise les images en jetons visuels. L'encodeur du transformateur fonctionne sur des patchs de taille fixe dans une image et

les intègre séparément des données de position avant de les transmettre au modèle CNN principal. La couche d'auto-attention permet d'intégrer globalement des informations sur une image. Le modèle apprend également sur les données d'entraînement pour coder l'emplacement relatif des patchs dans une image.

17 Conclusion

Les réseaux de neurones convolutions (CNN) et les transformateurs visuels (ViT) sont deux approches différentes très utilisées dans le domaine de la vision par ordinateur. Les deux approches ont leurs avantages et leurs limites, et le choix entre l'une ou l'autre dépendra des besoins spécifiques.

Chapitre 3

Conception et Implementation

1 Introduction

Après avoir présenté les principes fondamentaux des systèmes de détection de piétons et de l'intelligence artificielle dans les chapitres précédents, et approfondi les détails du réseau CNN et des transformateurs de vision dans le chapitre deux, nous allons dans ce chapitre developper l'architecture du système de détection de piétons proposé et son implementation avec divers tests effectués sur des images et des vidéos.

2 Architecture géneérale du systeḿe

Notre systeme se compose de 3 modules principaux :

- Module de pretraitement.
- Module d'apprentissage du modèle .
- Module de detection des piétons.

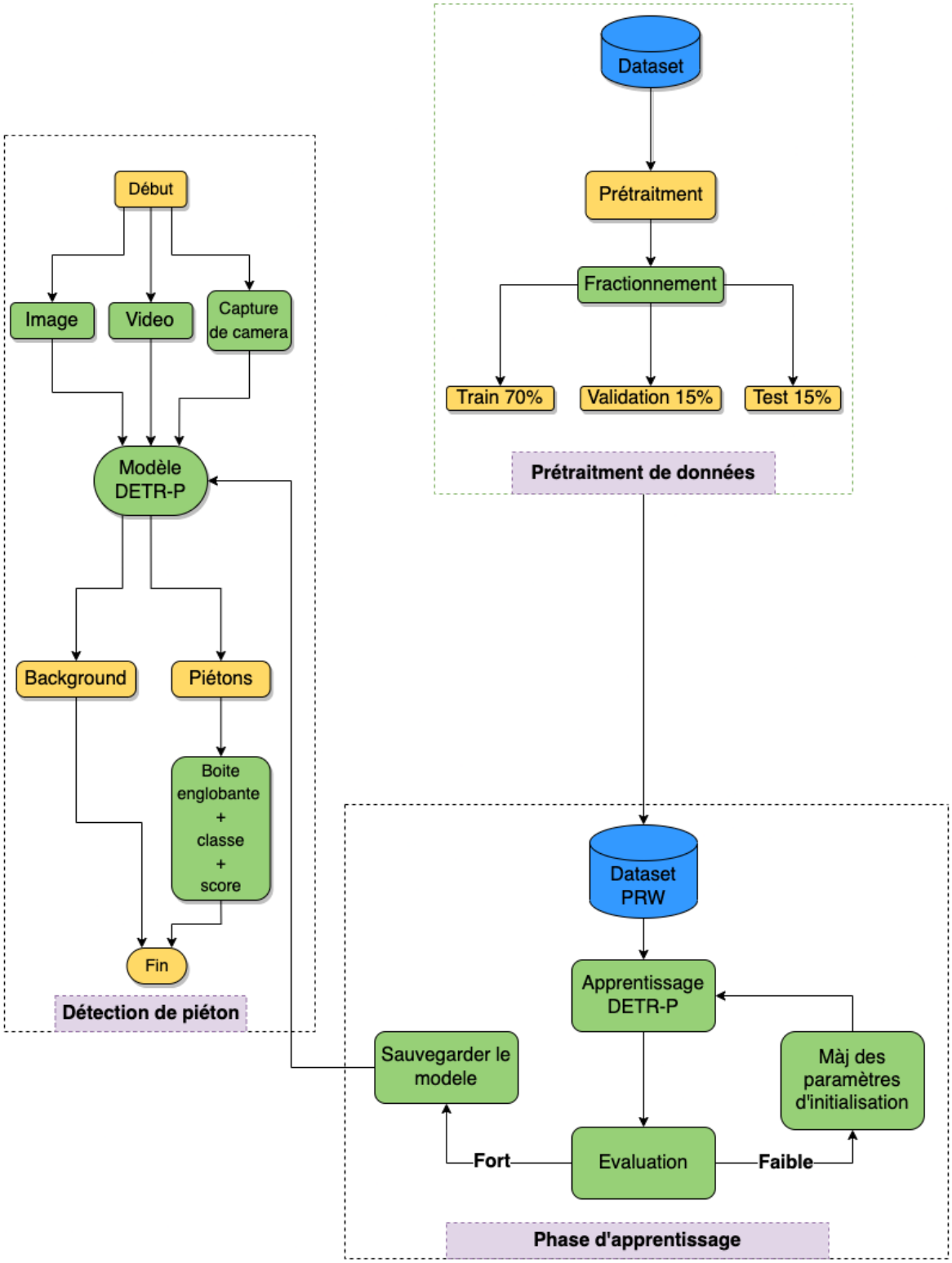


FIGURE 3.1 – Architecture de notre systeme

3 Architecture du model DETR-P

Le modèle que nous avons adopté dans l'architecture de notre système est un modèle basé sur une architecture hybride CNN et transformateur proposée à l'origine par Nicolas Carion et al. L'article intitulé "End-to-End Object Detection with Transformers" [25]. L'architecture est représentée par la figure 3.2. Il comprend trois composants principaux, un réseau de neurones convolutif (CNN) pour extraire une représentation de caractéristiques compacte, un transformateur encodeur-décodeur et un simple réseau de neurones à propagation directe (feed forward network) qui effectue la prédiction finale de détection.

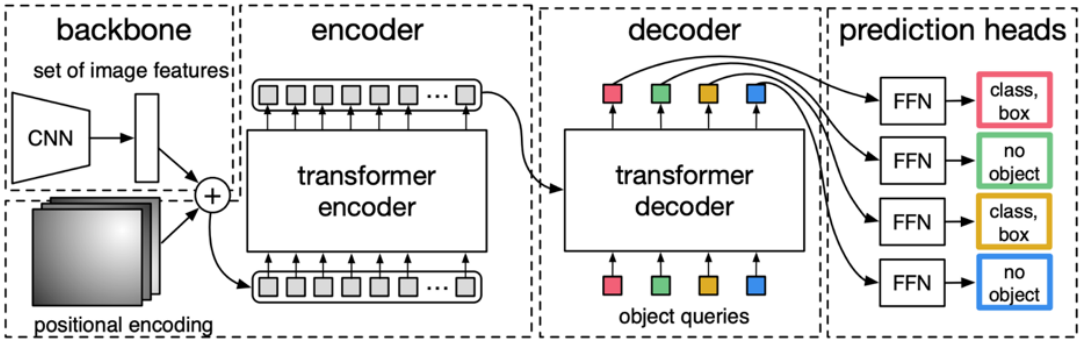


FIGURE 3.2 – Architecture générale du DETR

3.1 Backbone CNN

Ce composant est responsable de l'extraction des caractéristiques visuelles à partir de l'image d'entrée. Il s'agit généralement d'un réseau de neurones convolutif pré-entraîné ResNet-50. Le backbone CNN capture les informations visuelles à différentes échelles et niveaux d'abstraction à travers des couches de convolution et de pooling. À partir de l'image initiale (C x H0 x W0) avec C=3 canaux de couleur, l'image d'entrée est passée à travers le CNN backbone. Cette étape produit une carte des caractéristiques avec des dimensions C x H x W, où C représente le nombre de canaux et H, W représentent respectivement la hauteur et la largeur de la carte des caractéristiques. La taille de la carte des caractéristiques est généralement réduite de 32 par rapport à la taille originale de l'image. Le choix spécifique de diviser par 32 provient de la configuration de la structure du backbone. puisque le backbone a cinq couches de pooling, ce qui réduit la taille spatiale des caractéristiques par un facteur de 2 à chaque étape de pooling. Par conséquent, la taille des caractéristiques est réduite de manière exponentielle, d'où la division par 32..

Les valeurs typiques utilisées sont C = 2048 et H=H0/ 32, W= W0/ 32. La dernière couche de sortie du réseau aura 2048 canaux d'activation, indiquant qu'il y a 2048 canaux d'activation dans la carte des caractéristiques.

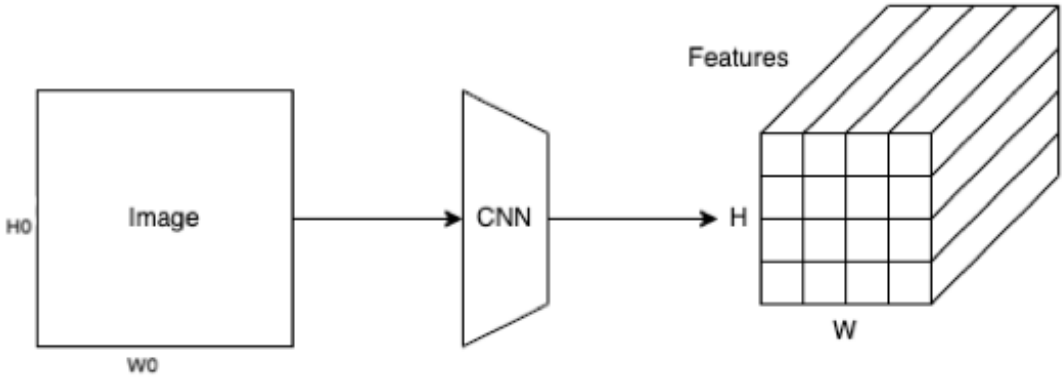


FIGURE 3.3 – Extraction des caractéristiques

Ensuite, DETR applique une couche de convolution de taille 1x1. Cette couche de convolution effectue une transformation linéaire sur chaque emplacement spatial de la carte des caractéristiques, réduisant ainsi le nombre de canaux de 2048 à 256. Cette réduction de dimension permet de compresser les informations de la carte des caractéristiques tout en conservant les caractéristiques importantes.

En conséquence de cette opération, chaque emplacement de cellule de grille dans la carte des caractéristiques correspond désormais à un vecteur de 256 valeurs. Ces vecteurs capturent les caractéristiques visuelles importantes de l'image à différents emplacements spatiaux. Ensuite, DETR aplati la carte des caractéristiques de taille H x W en un vecteur de caractéristiques de taille HW. Cela signifie que chaque position dans la carte des caractéristiques est transformée en un vecteur de caractéristiques individuel. Ensuite, DETR ajoute un encodage de position à chaque vecteur de caractéristiques.

L'aplatissement de la carte des caractéristiques permet de traiter chaque position de manière indépendante. L'encodage de position est une représentation numérique qui indique la position relative de chaque vecteur de caractéristiques dans l'image. En ajoutant des encodages de position à chaque vecteur de caractéristiques, DETR permet au modèle de capturer à la fois les informations visuelles et les informations de position de chaque objet dans l'image. Cela est essentiel pour la localisation précise des objets lors de l'étape de détection d'objets. la sortie est ensuite transmise à l'encodeur du transformateur en entrée.

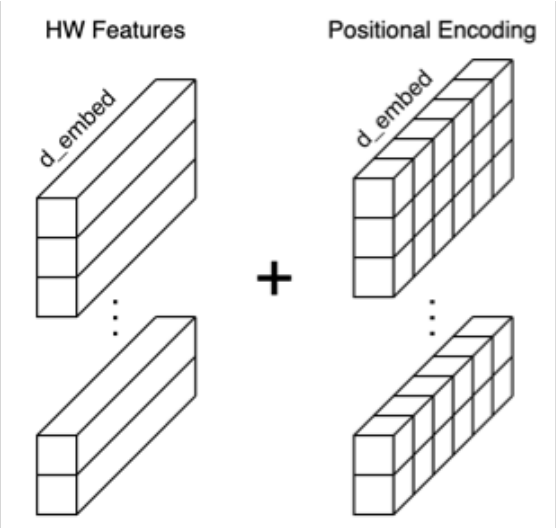


FIGURE 3.4 – Encodage de position

3.2 Transformateur Encodeur-Décodeur

Le modèle DETR utilise l'architecture de Transformateur pour capturer les informations contextuelles globales et effectuer la détection d'objets de manière parallélisable.

3.2.1 Encodeur

L'encodeur prend les caractéristiques visuelles extraites par le backbone CNN et les encode en utilisant une pile de couches d'encodeur transformateur. Chaque couche d'encodeur comprend des mécanismes d'auto-attention multi-têtes , des réseaux de neurones à propagation directe (feed forward neural networks) par position, FFN sont également connus sous le nom de perceptrons multicouches (MLP) et des couche de Connexions résiduelles et normalisation. L'encodeur capture les relations et les dépendances entre les différentes caractéristiques visuelles.

3.2.2 Décodeur

Le décodeur se focalise sur les caractéristiques encodées et génère un ensemble de requêtes d'objet (object queries). Chaque requête d'objet représente un potentiel objet dans l'image. Le décodeur comprend des couches de décodeur transformateur qui effectuent des opérations d'auto-attention multi-têtes sur les requêtes d'objet et les caractéristiques vi-

suelles encodées. Le décodeur affine progressivement les requêtes d'objet et génère les prédictions finales.

3.3 Réseau de neurones à propagation directe (Feed Forward Network)

Ce composant est utilisé pour effectuer la prédiction finale de détection. La prédiction finale est calculée à l'aide d'un perceptron à 3 couches avec une fonction d'activation ReLU et une dimension cachée d. Ce réseau à propagation directe (FFN) prédit les coordonnées normalisées du centre, la hauteur et la largeur de la boîte englobante par rapport à l'image d'entrée. Une couche linéaire est utilisée pour prédire l'étiquette de classe en utilisant une fonction softmax.

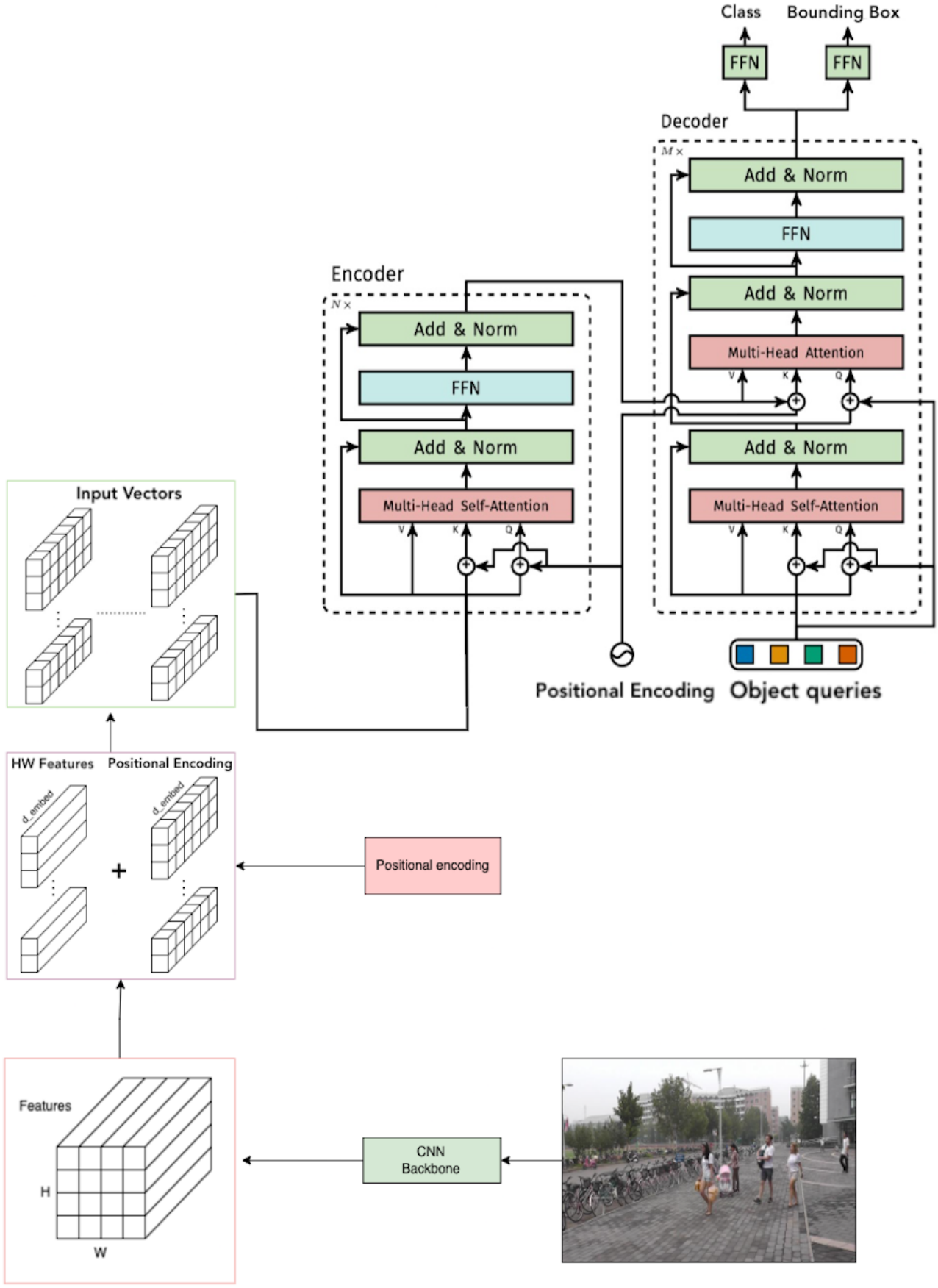


FIGURE 3.5 – Architecture détaillée du DETR-P

4 Configuration du modèle

La configuration du modèle DETR comprend plusieurs composants qui définissent son architecture et son comportement.

- Le backbone du modèle est défini comme étant ResNet50, une architecture de réseau de neurones convolutifs populaire reconnue pour ses excellentes capacités d'extraction de caractéristiques.

- L'entrée du modèle se compose d'images avec trois canaux de couleur (**num_channels=3)**.

- L'encodeur est construit à l'aide de couches de transformeur, avec **encoder_layers** défini sur 6. Chaque couche de l'encodeur comprend un mécanisme d'auto-attention multi-têtes avec **encoder_attention_heads= 8** têtes, permettant au modèle de capturer les relations et les dépendances entre les caractéristiques visuelles. Les réseaux de neurones à propagation directe (FFN) de l'encodeur ont une dimension de **encoder_ffn_dim=2048**.

- Le décodeur est composé de couches de transformeur, avec **decoder_layers** défini sur 6. Le décodeur utilise également une auto-attention multi-têtes avec **decoder_attention_heads = 8** têtes. Les FFN du décodeur ont une dimension de **decoder_ffn_dim=2048**.

- La sortie de l'encodeur est alimentée dans le décodeur. La taille de la sortie du modèle est déterminée par **d_model=256**. Une régularisation de dropout est appliquée avec un taux de **dropout=0.1** pour éviter le surapprentissage.

- La fonction d'activation utilisée dans l'ensemble du modèle est ReLU (**activation_function="relu"**).

```
def __init__(
    self,
    use_timm_backbone=True,
    backbone_config=None,
    num_channels=3,
    num_queries=100,
    encoder_layers=6,
    encoder_ffn_dim=2048,
    encoder_attention_heads=8,
    decoder_layers=6,
    decoder_ffn_dim=2048,
    decoder_attention_heads=8,
    encoder_layerdrop=0.0,
    decoder_layerdrop=0.0,
    is_encoder_decoder=True,
    activation_function="relu",
    d_model=256,
    dropout=0.1,
    attention_dropout=0.0,
    activation_dropout=0.0,
    init_std=0.02,
    init_xavier_std=1.0,
    auxiliary_loss=False,
    position_embedding_type="sine",
    backbone="resnet50",
    use_pretrained_backbone=True,
    dilation=False,
```

FIGURE 3.6 – Configuration du modèle

5 Implémentation

5.1 Matériel

Nous avons utilisé un ordinateur avec les spécifications suivantes pour notre projet :

- RAM : 16 Go

- Processeur : Intel Core i7

- GPU : NVIDIA GeForce GT 750M 2 GB

Ces composants matériels ont été utilisés pour faciliter l'exécution fluide et le traitement efficace de notre application.

5.2 Logiciel

1. **Environnement Google Colab**

   Google Colab ou Colaboratory est un service cloud, offert par Google (gratuit), base

sur 'Jupyter Notebook' et destine a la formation et a la recherche dans l'apprentissage automatique. Cette plateforme permet d'entrainer des modeles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur l'ordinateur a l'exception d'un navigateur. [w17]

2. **Le Software PyCharm CE**

   PyCharm CE fait référence à PyCharm Community Edition, qui est un environnement de développement intégré (IDE) gratuit et open-source spécifiquement conçu pour la programmation Python. PyCharm est développé par JetBrains, une entreprise de développement de logiciels réputée pour la création d'IDE puissants.[w18]

3. **Langage de programmation : Python**

   Python est un langage de programmation de haut niveau, interprété et orienté objet. Il est très demandé par une large communauté de développeurs et de programmeurs. Python est un langage simple et facile à apprendre. La bibliothèque Python est disponible pour la plupart des plates-formes et peut être redistribuée gratuitement. [w19]

5.3 Bibliotheques utilisees

1. **Open-CV**

   OpenCV (Open Source Computer Vision Library) est une bibliotheque logicielle open source de vision par ordinateur et d'apprentissage automatique. OpenCV a ete construit pour fournir une infrastructure commune pour les applications de vision par ordinateur et pour accelerer l'utilisation de la perception artificielle dans les produits commerciaux. Etant un produit sous licence BSD, OpenCV permet aux entreprises d'utiliser et de modifier facilement le code. [w20]

2. **Numpy**

   NumPy est le package fondamental pour le calcul scientifique en Python. Il s'agit d'une bibliothèque Python qui fournit un objet tableau multidimensionnel, divers objets dérivés (tels que des tableaux masqués et des matrices) et un assortiment de routines pour des opérations rapides sur des tableaux, y compris mathématiques, logiques, manipulation de forme, tri, sélection, E/S (Entrée/Sortie), transformées de Fourier discrètes, algèbre linéaire de base, opérations statistiques de base, simu-

lation aléatoire. . . etc.[w21]

3. **Os**

   Le module OS en Python fournit des fonctions pour interagir avec le système d'exploitation. Le système d'exploitation fait partie des modules utilitaires standard de Python. Ce module fournit un moyen portable d'utiliser les fonctionnalités dépendantes du système d'exploitation. Les modules *os* et *os.path* incluent de nombreuses fonctions pour interagir avec le système de fichiers telles que la création et la suppression d'un répertoire (dossier), la récupération de son contenu, la modification et l'identification du répertoire courant, etc.[w22]

4. **Torch**

   Torch est un framework d'apprentissage automatique open-source qui offre une bibliothèque puissante pour le calcul tensoriel et la création de modèles de réseaux de neurones. Il est largement utilisé pour la recherche et le développement dans le domaine de l'apprentissage automatique.[w23]

5. **Pytorch_lightning**

   PyTorch Lightning est une bibliothèque open-source qui facilite le développement et l'entraînement de modèles de réseaux de neurones avec PyTorch. Il fournit une interface simplifiée et abstraite qui permet aux développeurs de se concentrer sur la conception du modèle et l'expérimentation, en automatisant une grande partie du code répétitif lié à l'entraînement et à l'évaluation des modèles.[w24]

6. **Tkinter**

   Tkinter est une bibliothèque standard de Python qui permet de créer des interfaces graphiques (GUI - Graphical User Interfaces) pour des applications de bureau. Tkinter est basé sur le toolkit Tk, qui est un ensemble d'outils pour la création d'interfaces graphiques.[w25]

5.4 Base d'apprentissage

Nous avons utilise la base 'PRW (Person Re-identification in the Wild) Dataset[11]'. Cette base contient 2 dossiers principaux. Le dossier "frames", il contient 11816 images, et le dossier "annotations" contient 11816 fichiers mat en correspondance avec chaque image dans le dossier "frames". Toutes les cases annotées sont des piétons. Chaque fichier MAT

enregistre la position de la boîte englobante dans le cadre. Pour l'adapter avec notre reseau, nous avons fait quelques modifications sur cette base.

FIGURE 3.7 – Exemple d'images de la base de données

5.5 Pretraitement de donneées

5.3.1 Redimensionnement des images

Pour améliorer l'efficacité de calcul et assurer une cohérence dans le traitement, les images de l'ensemble de données ont été redimensionnées 600*400. Le redimensionnement consiste à ajuster les dimensions des images à une taille prédéterminée. Cette étape permet d'obtenir une uniformité dans les tailles d'images d'entrée, ce qui est essentiel pour que le modèle DETR effectue une détection d'objets précise et efficiente.

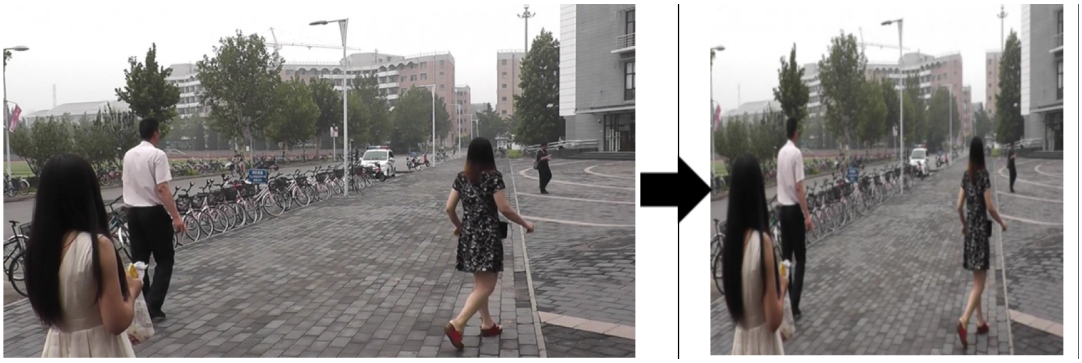


FIGURE 3.8 – Redimensionnement des images

5.3.2 Conversion des fichiers d'annotations

Afin de rendre les fichiers d'annotations compatibles avec le modèle DETR, une conversion a été effectuée, passant du format original '.mat' au format COCO JSON plus couramment utilisé. Cette conversion facilite l'intégration sans faille des annotations dans le pipeline du modèle DETR.

5.3.3 Fractionnement de la base de données

Afin d'évaluer les performances du modèle DETR, les données a été divisé en trois ensembles : l'ensemble d'entraînement 70% , l'ensemble de validation 15% et l'ensemble de test 15%. L'ensemble d'entraînement est utilisé pour entraîner le modèle, l'ensemble de validation est utilisé pour l'ajustement des hyper-paramètres et l'évaluation du modèle pendant l'entraînement, et l'ensemble de test est utilisé pour évaluer les performances finales du modèle entraîné.

5.6 Apprentissage et test

5.4.1 Apprentissage

Pour l'apprentissage de notre modèle nous avons suivi les étapes suivantes :
— La préparation de l'environnement de l'exécution Google Colab pour un mode GPU.
— La preparation de la base d'apprentissage.
— L'importation des bibliotheques necessaires.
— L'entrainement du modele.
— L'evaluation du modele.
— le sauvegarde du modele obtenue.
Après plusieurs test et changements des configuration du model, Nous avons obtenu les configurations suivantes :

- Optimizer : "Adam".
- Batch size : 8.
- Epochs : 30.

5.4.2 Estimation des performances du modèle

L'évaluation d'apprentissage est représenté sur la figure 3.9 par le courbe de pertes(Loss) obtenues sur les données d'apprentissage et de validation. Sur cette courbe nous constatons que la perte minimale obtenue aprés 30 'époch' est de 9% pour les données d'entrainement (train dataset) et de 10% pour les données de validation (validation dataset).

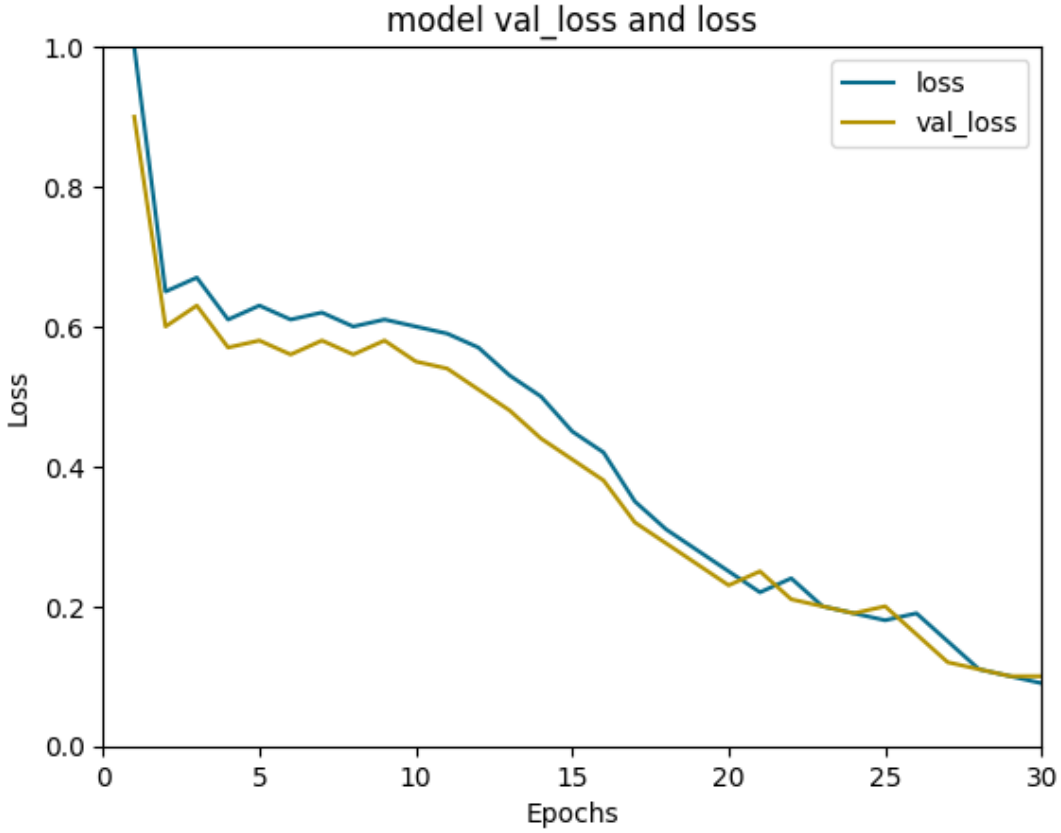


FIGURE 3.9 – Graphe de "loss" et "validation loss"

Pour évaluer la precision du modèle, nous avons utilisé la métrique suivante :

- nous avons utilisé la métrique IOU sur l'ensemble de données de test. Il mesure le chevauchement entre la boîte englobante prédite et la boîte englobante de vérité terrain.IoU est calculé comme le rapport de la zone d'intersection entre les deux boîtes englobantes à la zone d'union. Une boîte englobante prédite est considérée comme une bonne détection si son IOU avec la boîte englobante de vérité terrain dépasse un seuil prédéfini, généralement entre 0,5 et 0,7.

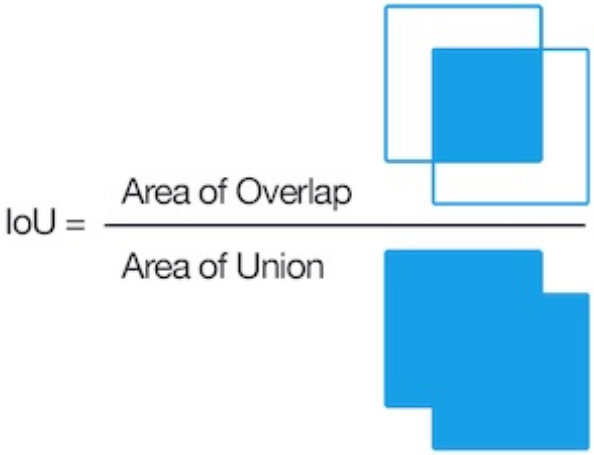


FIGURE 3.10 – Équation de métrique IoU

- Les valeurs de précision moyenne (Average Precision, AP) représentent la performance du modèle de détection d'objets sur une plage de seuils IoU (Intersection over Union) spécifiés.

$$AP = \frac{\sum_{i=1}^{n}(P(i) \cdot \Delta R(i))}{R_{\text{total}}} \tag{3.1}$$

les résultats obtenus sont illustrés dans le tableau 3.1 . Nous avons obtenu une précision de 0,911 à un seuil IOU de 0,50. De même, à un seuil IoU de 0,75, le modèle atteint une précision de 0,613. Il convient de noter qu'un score IoU supérieur à 0,5 est généralement considéré comme une prédiction réussie, et les performances du modèle dépassent ce seuil. De plus, le modèle démontre une précision louable lors de la détection d'objets de taille moyenne et grande, avec des scores de précision respectifs de 0,638 et 0,75.

IOU	Area	Average Precision (AP)
[0.50:0.95]	All	0.560
[0.50]	All	0.911
[0.75]	All	0.613
[0.50:0.95]	Small	0.363
[0.50:0.95]	Medium	0.638
[0.50:0.95]	Large	0.756

TABLE 3.1 – Tableau des résultats

5.7 Quelques tests sur des images aleatoires

Le tableau 3.2 illustre quelques tests obtenus sur des images d'internet. les tests sont effectués sur des images de complexité variable. Nous avons commencé avec des images contenant un nombre restreint de piétons (2 à 5) dans les images 1, 2, 3 et 4, puis nous avons augmenté le nombre de piétons dans les images 5, 6, 7 et 8, qui présentent également une forte présence d'occultation.

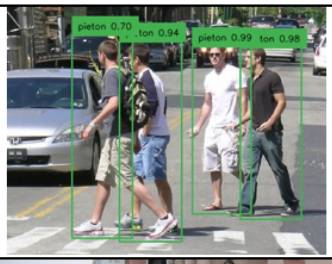
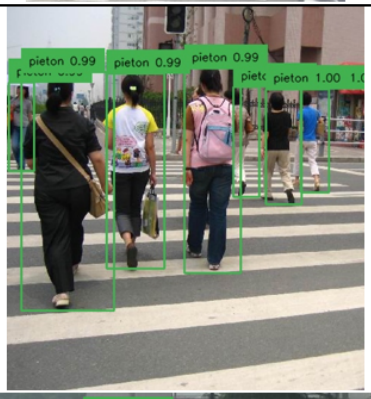
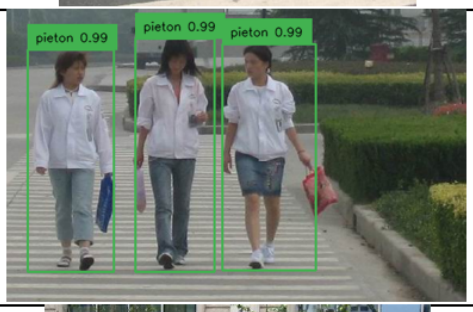
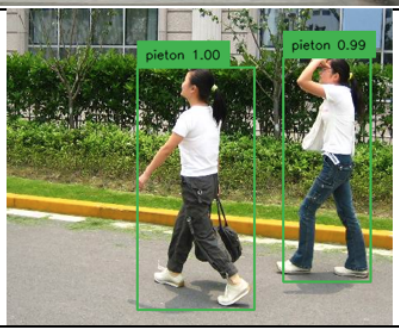
	Originale	Resultat
1		
2		
3		
4		

TABLE 3.2 – Illustration de quelques tests sur des images.

5.8 Quelques tests sur des vidéos

Le tableau 3.2 illustre quelques tests obtenus sur des vidéo filmées par nous et d'internet.

Vidéo	Frame	Résultat
Vidéo du net		

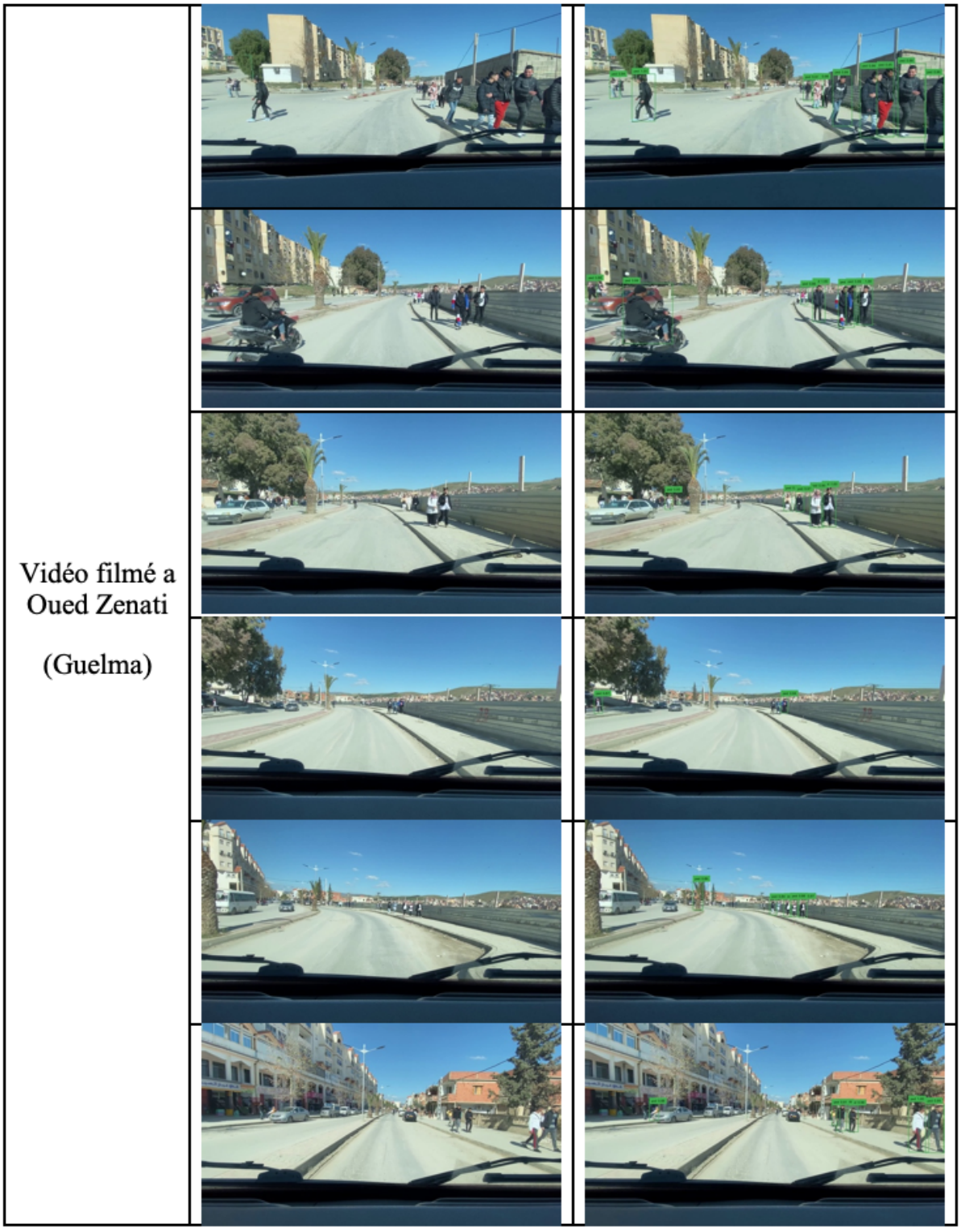


TABLE 3.3 – Illustration de quelques tests sur des vidéo

Notre système réussit dans la majorité des cas à détecter correctement les piétons quelque soit leurs positions avec des précisions acceptable. Il faut noter que l'échec de notre système est signalé sauvent dans les cas où un piéton est caché par un autre ou lorsque le piéton

est trop loin de l'objectif de la camera.

5.9 Interface du systéme

L'interface principale de notre système est illustrée par la figure 3.10.

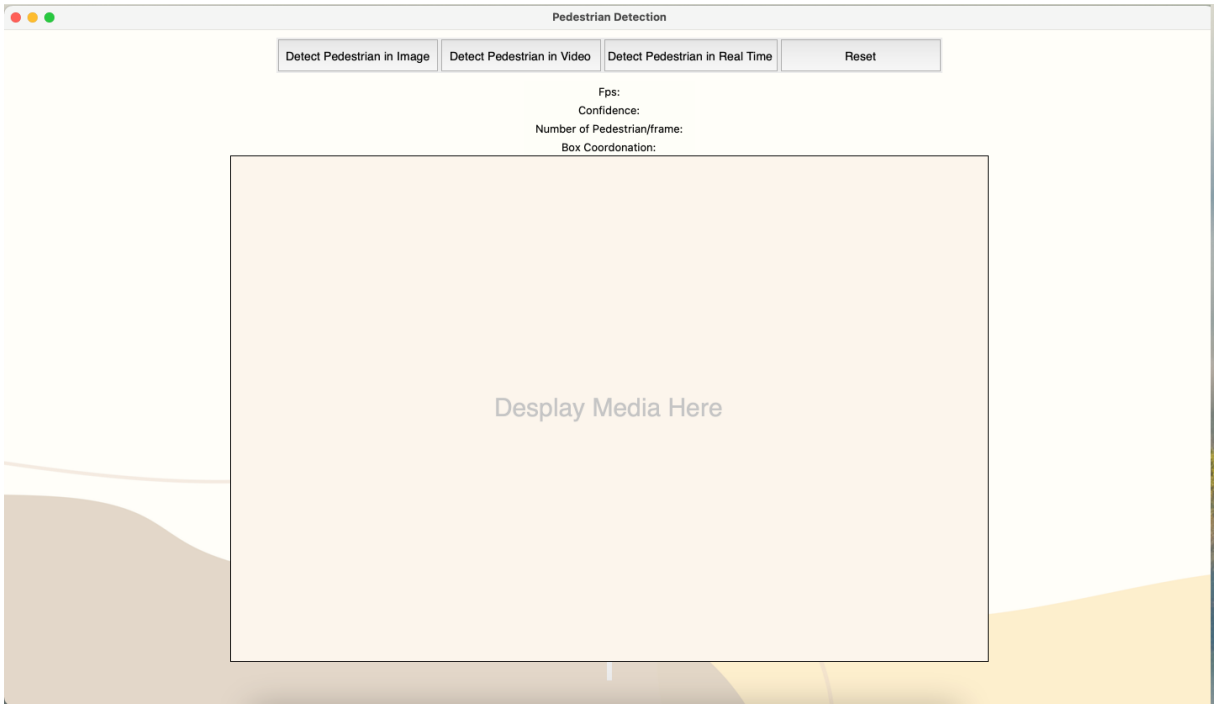


FIGURE 3.11 – Interface principale de notre système.

- **Bouton "Detect Objects in Image" :** permet de charger une image.

- **Bouton "Detect Objects in Video" :** permet de charger une vidéo.

- **Bouton "Detect Objects in Real time" :** permet d'ouvrir la caméra et faire détection en temps réel.

- **Bouton "Reset" :** permet de réinitialiser les champs de saisie ou de restaurer l'état initial de l'interface.

FIGURE 3.12 – Exemple de détection sur image.

FIGURE 3.13 – Exemple de détection sur vidéo.

Fenêtre supplémentaire

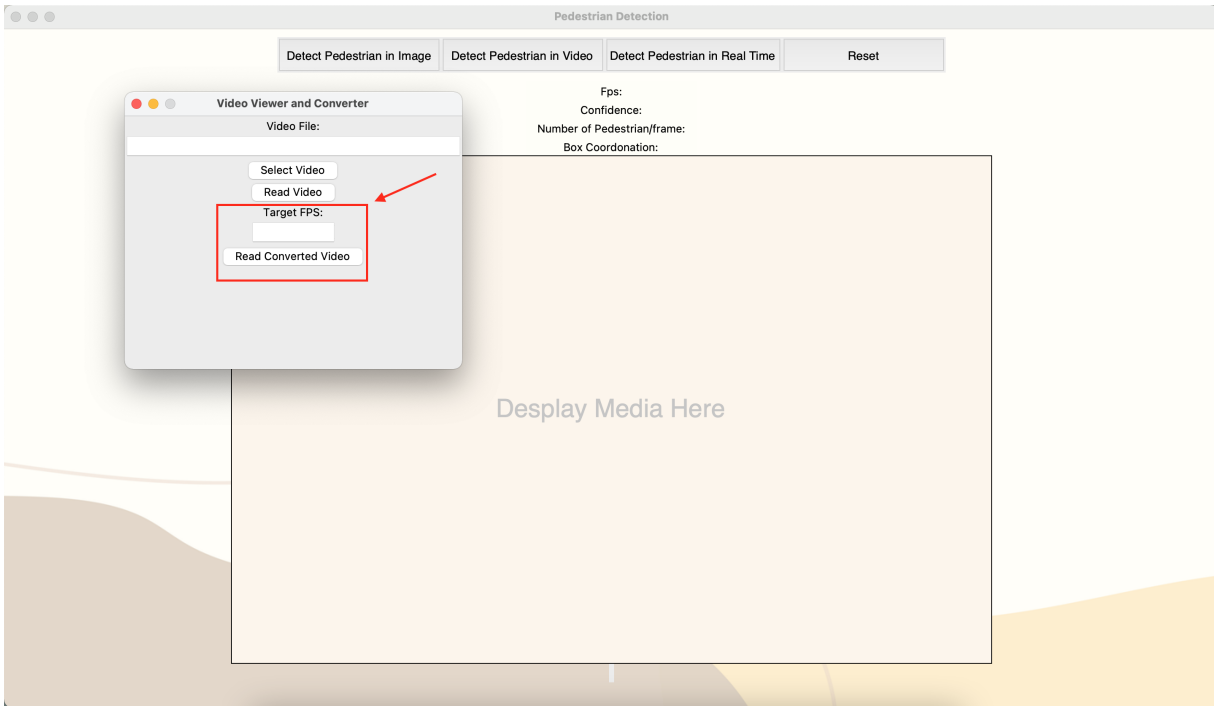


FIGURE 3.14 – Fenêtre supplémentaire pour changer le nombre de FPS

- **Champ "Target FPS" :** champ pour entrer le nombre d'images par seconde.
- **Boutton "Read Converted Video" :** permet de convertir la vidéo en un nombre spécifique de frame par seconde et de l'afficher.

6 Conclusion

Nous avons mis en oeuvre un système de detection des piétons sur des images ou des videos dont les performances ont été testés sur différent types d'images et videos a complexité variable. Les résultats obtenus sont très promoteurs mais peuvent être optimisées en utilisant une grande quantité d'images et en prolongeant la phase d'apprentissage, et aussi en ameliorant la qualite du 'hardware' comme (GPU).

Conclusion

Au cours des dernières années un accroissement rapide des accidents de la route a été noté. Avec le grand développement dans le domaine de l'intelligence artificielle ce phénomène peut être réduit en créant un system pour la détection des piétons dans les zones routières afin d'alerter le conducteur a temps pour prendre ses précautions sécuritaires.

Notre projet s'est concentré sur le développement d'une application permettant de détecter les piétons dans les zones routières. Dans le système proposé nous avons utilisé un réseau profond basé sur une combainison de CNN et de transformateur nomé DETR mis en oeuvre pour la detection des objets, que nous avons ré-entrainer pour detecter uniqument les piétons. DETR-P est une adaptation du modèle DETR pour se concentrer uniquement sur la détection des piétons.Les performances ont été testés sur différent types d'images et videos a complexité variable. Les résultats obtenus sont très promoteurs mais peuvent être optimisées.

En ce qui concerne l'avenir, il existe plusieurs pistes de recherche à explorer. Une direction potentielle consiste à affiner les performances du système en affinant le modèle sur des ensembles de données plus grands et plus diversifiés, en améliorant la qualité du matériel de traitement (GPU) et en permettant aux modèles de s'entraîner plus longtemps.

De plus, étendre l'application pour inclure d'autres aspects critiques de la sécurité routière, tels que la détection d'autres objets et la reconnaissance des comportements suspects des véhicules, permettrait de proposer une solution complète pour réduire les accidents et sauver des vies. Cette expansion de l'application offrirait une protection sur les routes, en détectant les différents types d'obstacles et de comportements potentiellement dangereux.

Enfin, cette application peut être considérée comme un premier pas vers la réalisation d'un système de détection des piétons efficace, ouvrant la voie à de futures améliorations et développements dans le domaine de la sécurité routière.

Bibliographie

[1] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Caltech pedestrians, Jun 2009.

[2] Markus Enzweiler and Dariu M. Gavrila. Monocular pedestrian detection : Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12) :2179–2195, December 2009.

[3] Shuai Shao, Zijian Zhao, Boxun Li, Yiming Dai, Liang Chen, Gang Xu, Yanwei Zhang, Xiangyu Wang, and Ying Wu. Crowdhuman : A benchmark for detecting human in a crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 368–383. Springer, 2018.

[4] Markus Braun, Sebastian Krebs, Fabian B. Flohr, and Dariu M. Gavrila. Eurocity persons : A novel benchmark for person detection in traffic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.

[5] Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi, and In So Kweon. Multispectral pedestrian detection : Benchmark dataset and baselines. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. "https://soonminhwang.github.io/rgbt-ped-detection".

[6] Xuan-Thuy Vo, Van-Dung Hoang, Duy-Linh Nguyen, and Kang-Hyun Jo. Pedestrian head detection and tracking via global vision transformer. In Kazuhiko Sumi, In Seop Na, and Naoshi Kaneko, editors, *Frontiers of Computer Vision*, Cham, 2022. Springer International Publishing.

[7] Mohamed Amine Marnissi, Ikram Hattab, Hajer Fradi, Anis Sahbani, and Najoua Essoukri Ben Amara. Bispectral pedestrian detection augmented with saliency maps using transformer. In *VISIGRAPP (5 : VISAPP)*, pages 275–284, 2022.

[8] Geonu Lee and Jungchan Cho. Stdp-net : Improved pedestrian attribute recognition using swin transformer and semantic self-attention. *IEEE Access*, 10 :82656–82667, 2022.

[9] Zaiming Sun, Chang'an Liu, Hongquan Qu, and Guangda Xie. Pvformer : pedestrian and vehicle detection algorithm based on swin transformer in rainy scenes. *Sensors*, 22(15) :5667, 2022.

[10] Katleho L Masita, Ali N Hasan, and Satyakama Paul. Pedestrian detection using r-cnn object detector. In *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6. IEEE, 2018.

[11] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, and Qi Tian. Person re-identification in the wild. *arXiv preprint arXiv :1604.02531*, 2016.

[12] Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Deep joint rain detection and removal from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1357–1366, 2017.

[13] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *IEEE transactions on circuits and systems for video technology*, 30(11) :3943–3956, 2019.

[14] Siyuan Li, Iago Breno Araujo, Wenqi Ren, Zhangyang Wang, Eric K Tokuda, Roberto Hirata Junior, Roberto Cesar-Junior, Jiawan Zhang, Xiaojie Guo, and Xiaochun Cao. Single image deraining : A comprehensive benchmark analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3838–3847, 2019.

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics : The kitti dataset. *The International Journal of Robotics Research*, 32(11) :1231–1237, 2013.

[16] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. Ua-detrac : A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 193 :102907, 2020.

[17] 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.

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861*, 2017.

[19] Claude Touzet. *les réseaux de neurones artificiels, introduction au connexionnisme.* Ec2, 1992.

[20] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53 :5455–5516, 2020.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929*, 2020.

[23] Khawar Islam. Recent advances in vision transformer : A survey and outlook of recent work. *arXiv preprint arXiv :2203.01536*, 2022.

[24] Javier Selva, Anders S Johansen, Sergio Escalera, Kamal Nasrollahi, Thomas B Moeslund, and Albert Clapés. Video transformers : A survey. *arXiv preprint arXiv :2201.05991*, 2022.

[25] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020 : 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

Webographie

[w1] STI. https ://www.techtarget.com/whatis/definition/intelligent- transportation-system.

[w2] Détection d'objets . https ://www.mathworks.com/discovery/object-detection.html

[w3] Détection de personnes.https ://www.cameralyze.co/blog/6-human-detection-applications-in-surveillance

[w4] Detection des piétons. https ://www.mathworks.com/discovery/object-detection.html

[w6] https ://blog.bismart.com/en/difference-between-machine-learning-deep-learning

[w7] types d'apprentissage.https ://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

[w8] Apprentissage par renforcement . https ://www.mathworks.com/discovery/reinforcement-learning.html

[w9] Structure globale d'un CNN.https ://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

[w10] Structure globale d'un CNN. https ://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[w11] Couche ReLu. https ://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn

[w12] https ://yannicksergeobam.medium.com/optimiser-les-hyperparam%C3%A8tres-dun-mo%C3%A8le-de-deep-learning-d65643141282

[w13] Optimiseur. https ://www.upgrad.com/blog/types-of-optimizers-in-deep learning/

[w14] https ://www.simplilearn.com/tutorials/machine-learning-tutorial/regularization-in-machine-learning#what_is_regularization_in_machine_learning

[w15] Epoque et Batch size.https ://datascientest.com/qu-est-ce-qu-un-epoch-en-machine-learning

[w16] Fonctions d'activation. https ://forum.huawei.com/enterprise/fr/qu-est-ce-que-la-

fonction-d-activation/thread/822591-100379

[w17] L'Environnement Google Colab. https ://ledatascientist.com/google-colab-le-guide-ultime/

[w18] Le Software PyCharm CE. https ://www.jetbrains.com/help/pycharm/quick-start-guide.html

[w19] Python. https ://www.python.org/doc/essays/blurb/

[w20] Opencv. https ://opencv.org/about/

[w21] Numpy. https ://numpy.org/doc/stable/user/whatisnumpy.html

[w22] Os. https ://www.javatpoint.com/python-os-module

[w23] Torch. https ://deepai.org/machine-learning-glossary-and-terms/torch

[w24] Pytorch_lightning. https ://pub.towardsai.net/pytorch-lightning-an-introduction-to-the-lightning-fast-deep-learning-framework-14325519cbaf

[w25] Tkinte. https ://www.tutorialspoint.com/python/pythonguiprogramming.