PROJET : DÉTECTION ET CLASSIFICATION DE PANNEAUX DE SIGNALISATION ROUTIÈRE

-

Université de technologie de Compiègne

-

Beijia Mao
Branchu Corentin

-

SY32

-

TD1 GroupeF

Semestre Printemps 2024

Sommaire

Introduction

Ce rapport comporte les différents éléments qui ont permis la mise en place d'un détecteur de panneau et de feu de signalisation dans des photos de mise en situation de rue en ville. Nous verrons dans un premier temps l'aspect de classification, points de départ nécessaire dans ce genre de projet. Puis nous verrons comment cette classification a été utilisée afin de détecter dans l'image les différents éléments que l'on recherche.
Nous proposons dans ce projet 2 méthodes de classification, la première plus avec des algorithmes classiques et la seconde avec algorithme de type réseau de neurones.

Classification classique

Choix du classifieur

Pour commencer nous avons besoin de trouver le meilleur classificateur étant données notre étude composé de panneau et feu de signalisation. Pour ce faire nous avons commencé par étudier implémenter les différents algorithmes classiques que sont, support vector machine (SVM), random forest, et d'autres. L'importance dans ces algorithmes est de pouvoir obtenir un classifieur pouvant gérer plusieurs classes afin de pouvoir implémenter chaque type de panneaux et de feux.
Nous avons entraîné ces modèles avec un set d'environ 1000 objets puis testé sur un set de 130 objets. Ces objets ont été ressortis depuis les images de mise en situation afin de ne pas avoir besoin pour le moment d'un algorithme de détection.
Les résultats sont présentés dans le tableau suivant:

Classificateur	Erreur
SVM	11%
Gradient boosting	14%
Random Forest	12%
(5,10,15) Nearest Neighbors	31%,30%,31%

Traitement des données

Le classificateur ayant obtenu les meilleurs résultats est le SVM, de ce fait c'est celui que nous allons utiliser dans le reste du projet pour la partie classique.
Nous avons dans un premier temps étudié les faux positifs ressorti à ce point. Il en est ressorti que nous faisions face à un surapprentissage au niveau de la classe des panneaux d'interdiction.

```
pred:  interdiction vs true:  ff
pred:  interdiction vs true:  obligation
pred:  fvert vs true:  frouge
pred:  interdiction vs true:  ceder
pred:  interdiction vs true:  stop
pred:  ff vs true:  obligation
pred:  interdiction vs true:  frouge
pred:  interdiction vs true:  ceder
pred:  interdiction vs true:  stop
pred:  ceder vs true:  danger
pred:  interdiction vs true:  frouge
pred:  interdiction vs true:  ceder
pred:  interdiction vs true:  forange
pred:  interdiction vs true:  obligation
pred:  danger vs true:  interdiction
pred:  interdiction vs true:  stop
pred:  interdiction vs true:  ceder
pred:  danger vs true:  forange
```

Figure 1: Faux positifs du premier entraînement

En effet, en regardant de plus près notre dataset nous avons observé une mauvaise répartition que nous allons réguler.

```
frouge: 90 objets
fvert: 109 objets
forange: 67 objets
ceder: 133 objets
obligation: 108 objets
danger: 162 objets
interdiction: 365 objets
stop: 100 objets
ff: 44 objets
```

Figure 2: Répartition des classes dans le set d'apprentissage

La classe interdiction sera donc réduite en supprimant directement les images depuis le dataset le tout pour arriver sur un total de 200 éléments. D'autre part, nous voyons dès maintenant que la classe "ff", représentant la classe négative ne présentant ni panneaux ni feux, est sous représenté, nous n'allons pas l'augmenter maintenant mais le feront une fois l'étape de détection commencer afin d'utiliser les faux positifs qu'elle générer.

```
pred:  danger vs true:  interdiction
pred:  danger vs true:  ff
pred:  interdiction vs true:  obligation
pred:  fvert vs true:  frouge
pred:  interdiction vs true:  stop
pred:  ff vs true:  obligation
pred:  interdiction vs true:  frouge
pred:  ceder vs true:  danger
pred:  interdiction vs true:  frouge
pred:  interdiction vs true:  ceder
pred:  interdiction vs true:  forange
pred:  fvert vs true:  obligation
pred:  interdiction vs true:  stop
pred:  fvert vs true:  interdiction
pred:  fvert vs true:  interdiction
pred:  interdiction vs true:  ceder
pred:  danger vs true:  forange
taux d'erreur:  12.977099236641221 %
```

Figure 3: Faux positifs avec moins d'interdiction

Le taux d'erreur n'as pas grandement diminué mais on peut voir que les faux positifs sont moins dû au panneaux interdiction. De plus, nous avons testé au fur et à mesure d'enlever

des panneaux d'interdiction, le taux d'erreur est descendu jusqu'à 10% mais avons tout de même décidé de continuer à en enlever.
Nous avons aussi constaté que la luminosité et la saturation des images varient considérablement en raison des conditions météorologiques, de l'heure de la prise de vue et des équipements utilisés. Par conséquent, nous avons utilisé des modifications aléatoires de la luminosité et de la saturation des images, ainsi que des renversements horizontaux aléatoires pour enrichir notre dataset.

Extraction des caractéristiques

Pour l'extraction de caractéristiques d'image, nous avons initialement choisi uniquement les caractéristiques de pixels, c'est-à-dire que nous avons directement aplati l'image en un vecteur unidimensionnel prêt à l'entraînement. Cependant, l'extraction des caractéristiques n'était pas satisfaisante et les performances de reconnaissance étaient faibles. Par conséquent, nous avons ajouté des caractéristiques HOG (Histogram of Oriented Gradients) en plus de celles des pixels. Cela permet de décrire la forme des objets par la distribution des directions de gradient locales. Les performances de reconnaissance étaient meilleures que celles obtenues en utilisant uniquement les caractéristiques de pixels.

```
pixel_features: 12288
hog_features: 1764
```

Figure 4: pixel_features VS hog_features

Toutefois, étant donné que le nombre de caractéristiques de pixels est bien supérieur à celui des caractéristiques HOG, le modèle est devenu plus sensible aux caractéristiques de couleur. Pour résoudre ce problème, nous avons ajouté un facteur de pondération aux caractéristiques HOG.

Voici les mesures de performance du modèle :

```
Val set Accuracy: 91.92%
Val set Recall: 91.92%
Val set Confusion Matrix:
 [[30  0  0  0  0  2  0  0  0]
 [ 0 38  0  0  0  0  0  0  0]
 [ 0  0  8  0  0  0  0  0  4]
 [ 0  0  0 20  2  2  0  0  2]
 [ 0  0  0  0  4  0  0  0  0]
 [ 0  0  0  0  0 82  0  0  6]
 [ 0  0  0  0  0  3 29  0  0]
 [ 0  0  0  0  0  0  0 28  0]
 [ 0  0  0  0  0  0  0  0  0]]
```

```
Class 0 AP: 1.0
Class 1 AP: 0.9999999999999999
Class 2 AP: 0.8720383986928105
Class 3 AP: 0.9678056764201028
Class 4 AP: 1.0
Class 5 AP: 0.9948852367291937
Class 6 AP: 1.0
Class 7 AP: 1.0
Class 8 has no positive samples in y_true. Skipping AP calculation.
Val set Mean Average Precision (mAP): 0.9793411639802634
```

Figure 5 & 6: les mesures de performance du SVM

Où les nombres de 0 à 8 représente successivement les classes 'ceder', 'danger', 'forange', 'frouge', 'fvert', 'interdiction', 'obligation', 'stop' ainsi que 'background'.
La classe 8 'background' est un échantillon négatif que nous n'avons pas besoin de reconnaître, il n'y a donc pas de balisage correspondant dans le set de validation (val).
Le taux d'erreur actuelle a donc diminué jusqu'à atteindre un taux de 8% ce que nous trouvons plus que convenable.

Classification par un réseau de neurones

Voyons à présent la méthode de classification avec un réseau de neurones.
Nous avons implémenté dans un premier temps un réseau similaire à celui de ResNet34.
Il se décompose de la manière suivante :

- Une couche de convolution initiale (7x7, 64 filtres, stride de 2, padding de 3), suivie de la normalisation par batch et de l'activation ReLU.
- Une couche de max-pooling (3x3, stride de 2, padding de 1).
- Quatre blocs résiduels (layer1 avec 3 BasicBlocks, layer2 avec 4 BasicBlocks, layer3 avec 6 BasicBlocks et layer4 avec 3 BasicBlocks).
  Chaque BasicBlock comprend deux couches de convolution 3x3, la normalisation par batch et l'activation ReLU, avec des couches de sous-échantillonnage si nécessaire.
- Une couche de global average pooling.
- Une dernière couche entièrement connectée pour la classification finale.

La raison pour laquelle nous avons choisi le réseau ResNet34 est que dans chaque bloc résiduel, l'entrée pré-convolution est utilisée directement comme partie de la sortie post-convolution. Cela permet d'éviter le problème de l'évanouissement des gradients et rend le réseau plus stable.

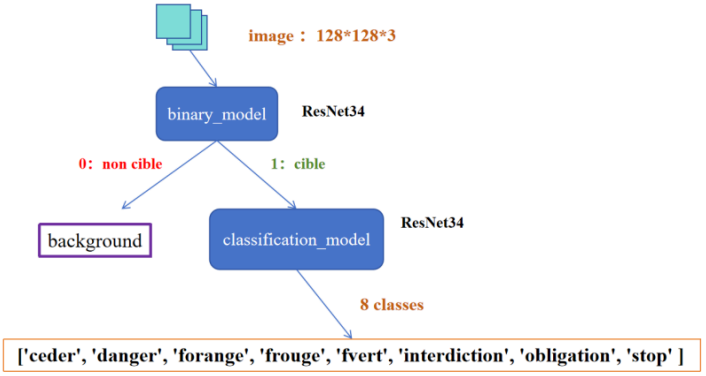

Figure 7: Structure du modèle à double réseau

En même temps, nous utilisons **un modèle à double réseau**. Nous entraînons le premier ResNet34 comme modèle binaire (binary_model), spécialisé dans la détection de la présence ou non de l'objectif recherché dans les images (panneaux de signalisation, feux de circulation). Le deuxième ResNet34, utilisé comme modèle de classification (classification_model), est spécialisé dans l'identification du type de chaque objectif dans les images filtrées par le premier modèle.

Comparé à l'utilisation d'un seul réseau pour la classification, cette approche réduit la probabilité de reconnaître un fond comme un objectif et améliore la précision de la reconnaissance de chaque type d'objectif.

Voici les mesures de performance du modèle :

```
Epoch 14/14, train Loss: 0.0426
Validation Loss: 0.1024
Confusion Matrix:
 [[  0   0]
 [  2 128]]
Accuracy total:  0.9846153846153847
Recall total:  [0.         0.98461538]
Accuracy per class:  0.9846153846153847
Recall per class:  [0.         0.98461538]
```

Figure 8: les mesures de performance du binary_model

```
Epoch 24/24, train Loss: 0.0460
Validation Loss: 0.1414
Confusion Matrix:
 [[15  0  0  0  0  1  0  0]
 [ 0 19  0  0  0  0  0  0]
 [ 0  0  6  0  0  0  0  0]
 [ 0  0  0 12  0  1  0  0]
 [ 0  0  0  0  2  0  0  0]
 [ 0  0  0  0  0 44  0  0]
 [ 0  0  0  0  0  0 16  0]
 [ 0  0  0  0  0  0  0 14]]
Accuracy total:  0.9846153846153847
Recall total:  [0.9375     1.         1.         0.92307692 1.         1.
 1.         1.        ]
Accuracy per class:  0.9846153846153847
Recall per class:  [0.9375     1.         1.         0.92307692 1.         1.
 1.         1.        ]
```

Figure 9: les mesures de performance du classification_model

Une autre méthode non testée par manque de temps est de baser notre réseau et son apprentissage à la fois sur classification comme actuellement mais aussi sur la détection à la manière des algorithmes Yolo ou SSD.

Pour ce faire, il est nécessaire de faire apparaître sur la dernière couche de notre réseau différents indicateurs. Le premier un élément booléen traduisant si un élément intéressant à été trouvé dans l'image (à la manière de notre modèle binaire plus haut). Un second élément sera évidemment les différentes probabilités pour chaque classe de notre étude de la même façon que notre classificateur actuel. Et enfin ressortir les coordonnées de deux coins de la boîte qu'aura détecté le réseau.

Cette méthode est plus compliquée à mettre en place que ce soit sur la méthode mais aussi sur les données d'apprentissage, dont il faudra modifier la composition de sorte à fournir non plus seulement des images de chaque classe mais les prendres dans une partie plus grande de la scène pour l'aspect de détection.

Détection

Voyons maintenant comment utiliser ces classificateurs afin de passer à une méthode de détection.

Sliding Window

Nous avons donc choisi d'implémenter la méthode de fenêtre glissante (sliding window) afin de parcourir l'image à la recherche d'éléments intéressants. Pour ce faire, nous avons choisi 3 tailles de boîtes, 128 par 128, 256 par 256 et 400 par 400. Testées les unes après les autres, elles extraient une partie de l'image qui est traitée par notre classificateur qui nous retourne la classe la plus probable. Une fois chaque fenêtre testée, nous sortons du lot les fenêtres les plus pertinentes en utilisant la méthode de filtre des non maximas et d'intersection sur union (IoU). Nous avons testé notre méthode dont voici le premier résultat.

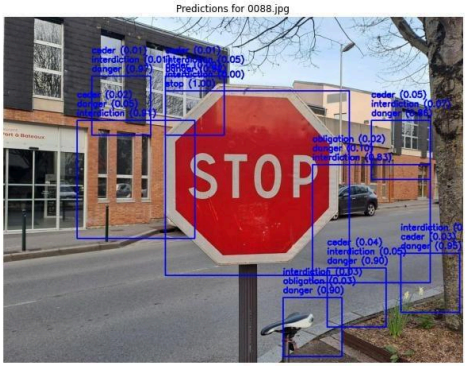

Figure 10: Premier test de détection

Comme vous pouvez le voir, le panneau stop a correctement été retrouvé ce qui est un début plutôt correct en revanche beaucoup d'éléments faux positifs ont été détectés ce qui n'est pas souhaitable. Cette mauvaise détection était prévisible comme nous l'avons fait remarquer dans la partie sur le traitement des données, le sous-entraînement de la classe ff (background) est sûrement à l'origine de ce problème. Nous avons utilisé ces différents faux positifs pour ajouter à notre set d'entraînement une quantité plus importante d'éléments de la classe ff.
Voyons maintenant le résultat après modification.

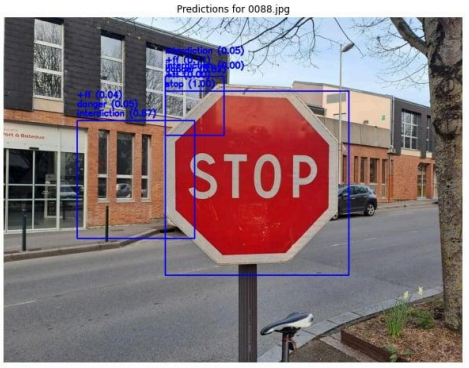

Figure 11: Test de détection sans sous apprentissage de la classe "ff" (background)

Le résultat est plutôt satisfaisant puisque nous avons réussi à éliminer la quasi-totalité des faux positifs au niveau du fond de notre image. Il ne reste que des détection au niveau de la partie rouge du panneau qui induit en erreur notre classificateur.

Nous avons ensuite testé d'autres images afin de voir d'autres pistes d'amélioration. Au niveau des feux de circulation nous nous sommes aperçus d'une limite de notre fenêtre glissante, les feux ne sont pas détectés dans leur hauteur mais de manière segmentée entre chaque rond de couleur, comme vous pouvez le voir sur la figure si dessous.

Figure 12: Détection des feux

Figure 13: Détection des feux avec SW dynamiques

En raison de la présence de différentes formes et proportions de cibles dans l'image. En fixant la taille de la fenêtre comme auparavant, la zone de cadrage ne serait pas très adaptée. C'est notamment le cas des feux de signalisation (rectangulaires) ou des panneaux de signalisation avec une certaine compression. Nous avons donc utilisé des fenêtres coulissantes dynamiques, qui peuvent être mises à l'échelle en fonction de la taille de l'image et de la taille du pas, ainsi que des fenêtres carrées, rectangulaires (1:2) et rectangulaires (2:1) pour la reconnaissance. Nous utiliserons donc 15 fenêtres de tailles différentes pour la reconnaissance. Bien sûr, cela sacrifiera le temps de traitement, mais le jeu en vaut la chandelle.

Recherche Sélective

Bien que les résultats soient bons, en raison du temps de traitement élevé du fait de l'utilisation de la fenêtre glissante dynamique, nous avons finalement opté pour la Recherche Sélective.
Voici ses principes de base :

- **Segmentation initiale** :L'image est segmentée en superpixels homogènes en couleur et texture.
- **Fusion hiérarchique** :Les superpixels sont fusionnés progressivement en régions plus grandes, basées sur la similarité de couleur, texture, taille et forme.
- **Génération de propositions** :À chaque fusion, des propositions de régions sont créées sous forme de rectangles englobant les régions fusionnées.
- **Diversité** :Plusieurs stratégies de fusion sont appliquées pour assurer une diversité de propositions en termes d'échelle et de couleur.
- **Classement et filtrage** :Les propositions sont classées et les moins pertinentes sont filtrées.

Figure 14: Sliding Window VS Recherche Sélective

La fenêtre coulissante dynamique prend 5 minutes pour reconnaître une seule image, tandis que la recherche sélective ne prend que 2 minutes pour reconnaître 7 images. Les avantages de la recherche sélective résident dans le fait qu'elle n'a pas besoin de faire glisser une fenêtre sur toute l'image. Elle peut fusionner les blocs en fonction de la similarité comparée par l'algorithme, générant ainsi rapidement des régions candidates pour les objets. Le tout en donnant des résultats qui ne sont pas dégradés.

Traitement supplémentaire des feux de signalisation

La détection de la couleur du feu peut aussi poser problème sur certaines images comme sur la figure 12, le feu vert n'est pas allumé mais est légèrement vert ce qui induit une erreur. Pour pallier ce problème, une possibilité pourrait être une méthode de post traitement au niveau des couleurs. Pour ce faire, nous utilisons des masques pour chaque couleur et regardons laquelle persiste avec le masque. L'exemple suivant montre la différenciation sur le feu de l'image au-dessus.

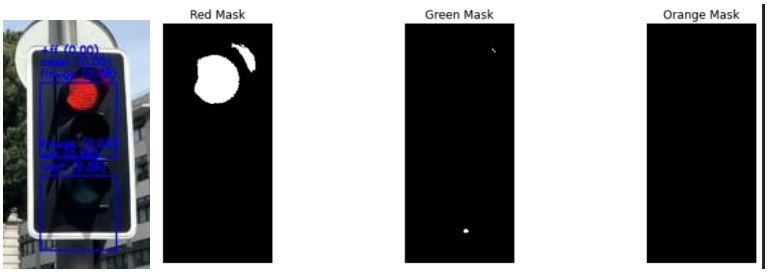

Figure 15: Masque appliqué à la détection des feux

Le résultat est celui escompté et nous permet bien de faire la différence entre chaque couleur de feu. Cela ne représente qu'un plus afin de s'assurer que la classification est correcte dans le cas ou 2 boîtes de feux auraient un score élevé. De ce fait, nous ne l'avons que testé mais pas ajouté à notre algorithme.

Conclusion

Ainsi pour conclure rapidement sur ce projet, nos résultats sur la partie classique sont plutôt convaincants. En continuant d'améliorer et d'harmoniser le set de données nous pouvons continuer d'améliorer les performances de nos algorithmes ce qui prend néanmoins du tout car c'est un processus difficilement automatisable. Nous pouvons aussi ajouter à notre post traitement de détection l'algorithme qui assure la bonne couleur des feux qui utilise des notions vues en début de semestre de SY32.
Du côté du réseau de neurones, comme vous pouvez le voir plus bas, les résultats sont plus faible ce qui est surprenant puisqu'en règle générale les réseaux de neurones sont plus

efficaces que les méthodes classiques.Une explication est sûrement le temps raccourci et le nombre de tests moins important que nous avons dédiés à cette partie.

2024-06-23 10:12:44	maobeiji	82.34	39.47	71.82	44.48	dl_resnet_test4	Télécharger
2024-06-22 20:54:36	maobeiji	93.01	25.96	84.86	29.04	ml_hog_svm_test1	Télécharger

Figure 16 : Résultats de chaque méthode utiliser