

# PROJET : DÉTECTION ET CLASSIFICATION DE PANNEAUX DE SIGNALISATION ROUTIÈRE

## Introduction

De nombreuses recherches sont menées pour alerter le conducteur sur les conditions routières. De nos jours, la plupart des véhicules sont construits avec des systèmes d'assistance de conduite ADAS (*Advanced Driver Assistance System*). Dans certaines voitures de série par exemple, les images des panneaux de signalisation sont captés grâce à la caméra fixée au devant du véhicule. Un algorithme de reconnaissance de caractères reconnaît ensuite le panneau de limitation de vitesse et l'affiche sur le tableau de bord du véhicule [[Automatic Traffic Sign Detection and Recognition: A Review](#)].

L'objectif de ce travail consiste à construire une base de données d'images de panneaux, et de concevoir un système de reconnaissance de ces panneaux. Le projet est fait en binôme, et les étapes à suivre ainsi que les livrables sont détaillés dans ce qui suit.

**NB :** Ce travail bénéficiera à l'équipe UTC participant au challenge étudiant UTAC : <https://www.utac.com/fr/challenge-utac/> !






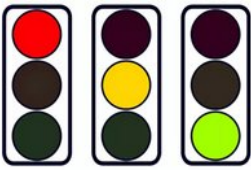
# Consignes et planning

## Phase 1. Collecte des images et effacement des métadonnées

Chaque étudiant devra fournir au moins 14 images de panneaux routiers.

Les signes retenus sont groupés par familles, réparties selon les informations du tableau suivant, pour obtenir un jeu de données équilibré :

**Tableau des données à collecter**

# Label classe	Type	Forme et couleurs	Nombre d'images nécessaires
"danger"	Danger		2
"interdiction"	Interdiction		2
"obligation"	Obligation		2
"stop"	Stop		2
"ceder"	Céder le passage		2
"frouge" "forange" "fvert"	Feux tricolores		1 de chaque couleur : rouge, orange, et vert (soit 3 au total)

<i>Image sans panneau ni feu</i>		1 (tout est perms)
----------------------------------	--	--------------------

Résolution et format des images : dimension de chaque côté comprise entre 400 et 1000 pixels, format JPG uniquement.

Conseil : essayez de faire en sorte que les panneaux ne soient pas trop petits.

La variation des conditions climatiques est possible (jour ensoleillé, journée sombre, soir, pluie, brouillard, etc).

**Pour adapter les images aux bonnes dimensions et enlever les métadonnées (avec application de la rotation, et pour enlever les données perso, GPS, ou permettant de réidentifier les auteurs), utiliser imagemagick et la commande suivante (à adapter de fonction de votre configuration) :**

**Linux :**

```
mkdir pret
convert *.JPG -auto-orient -strip -resize 1000x1000 pret/%03d.jpg
```

**Windows :**

```
mkdir pret
magick convert *.JPG -auto-orient -strip -resize 1000x1000 pret/%03d.jpg
```

## Phase 2. Annotation et anonymisation du contenu des images

Chaque image sera associée à un fichier CSV contenant les annotations, où chaque ligne définira la boîte englobante d'un objet, de la forme :

```
Coin haut-gauche x,Coin h-g y,Coin bas-droite x,Coin b-d y,Label classe
```

Les labels des classes sont donnés dans le tableau des types de panneaux à détecter.

Une image sans élément à détecter sera associée à un fichier CSV vide.

Chaque étudiant aura à annoter les images qui lui seront attribuées en utilisant les codes fournis. **Si jamais des images ne respectent pas les dimensions demandées, ou sont mal orientées, vous devrez d'abord les redimensionner ou les corriger.**

### Procédure d'annotation

Outils utilisés :

- labelImg : <https://github.com/HumanSignal/labelImg> (à cloner obligatoirement depuis le dépôt, les versions pip et conda n'étant pas tout fait à jour !)
- dépôt traffic\_sign avec des fichiers et scripts adaptés au projet : [https://gitlab.utc.fr/sy32/traffic\\_sign](https://gitlab.utc.fr/sy32/traffic_sign)

**1- Annotations :** Les images devront être annotées en utilisant labelImg avec le fichier de classes contenant les labels utilisés pour le projet : SY32\_classes.txt (inclut dans le dépôt traffic\_sign).

**Commande de lancement labelImg (à adapter de fonction de votre configuration) :**

```
python3 labelImg.py [IMAGE_PATH] SY32_classes.txt
```

### Anonymiser les visages, plaques d'immatriculation, et textes incrustés :

La classe "FF" correspond à « objets devant être floutés ». Il faut annoter avec cette classe les plaques d'immatriculation, les visages, et les textes incrustés (type infos de dashcam, pas les textes à l'intérieur des scènes), qui seront ensuite floutés par le script de traitement.

**2- Post-traitement :** Le dossier des images annotées devra porter le nom « images » et être mis dans le même répertoire de l'API fourni dans le dépôt traffic\_sign : traffic\_sign/pythonapi-main

Une fois le main exécuté, il y aura 3 nouveaux dossiers :

- *annotation\_csv* : contient les CSV des annotations.
- *treated\_images* : contient les images anonymisées.
- *visualized\_images* : contient les visualisations des annotations sur les images. Ces dernières peuvent servir à vérifier la bonne annotation des images.

**3- Données finales :** Ce sont les dossiers *treated\_images* et *annotation\_csv* qui devront être soumis ensemble pour intégrer le jeu de données final.

***Vérifiez que vous avez autant de fichiers images que de fichiers d'annotations !***

## Phase 3. Détection de panneaux par méthodes classiques (travail le plus important du projet)

### Base de données

La base d'images est disponible via le dépôt gitlab <https://gitlab.utc.fr/sy32/panneaux-p24/dataset/>

Le jeu de données est séparé en trois sous-ensembles classiques : *train* (80 % des données), *val* (10 % des données), et *test* (10 % des données). Le sous-ensemble de *test* ne sera fourni que deux semaines avant la fin du projet pour pouvoir vous auto-évaluer, avec vos algorithmes quasi-finalisés. Nous ne fournissons pas les annotations du sous-ensemble *test*, et vous ne devez pas vous entraîner avec, car c'est celui sur lequel vos résultats seront évalués.

En cas d'erreur d'annotation, vous êtes libres de la corriger et de la diffuser.

### Consignes

Chaque binôme est libre d'utiliser les méthodes classiques de détection vues en cours et disponibles dans les librairies existantes (scikit-learn par exemple) pour détecter et classifier les panneaux. Les méthodes d'apprentissage profond sont interdites pour cette partie.

Vous devez obtenir les meilleurs scores possibles, et expliquer vos progressions pour atteindre cet objectif. Pour cela, vous développerez vous-mêmes vos métriques et vos procédures d'améliorations. À vous de chercher quelles stratégies adopter, inspirez vous des concepts vus en cours et explorez d'autres concepts, le projet étant l'occasion d'aller plus loin.

### Page web d'auto-évaluation

Deux semaines avant la fin du projet, avec l'accès au sous-ensemble de *test*, un lien sera mis à votre disposition pour mesurer les performances de vos algorithmes sur les données de *test*.

## Phase 3-bis. Détection et classification de panneaux par méthodes d'apprentissage profond

En parallèle aux méthodes classiques, vous travaillerez à la détection avec un algorithme d'apprentissage profond. Vous êtes libres d'adopter n'importe quelle architecture réseau, mais vous devez **l'implémenter par vous-même** (pas de réemploi de sources existantes). Vous pouvez bien entendu vous baser sur une librairie d'apprentissage profond telle que pytorch ou tensorflow.

## Phase 4. Évaluation et barème

Chaque binôme sera tenu de soumettre les résultats de classification sur la plateforme de test fournie à la fin du projet (la page sera rendue accessible en même temps que les données de *test*).

### Livrables à déposer le 14 juin 2024

- **Rapport d'étude** : Chaque binôme est tenu de produire et remettre un rapport final expliquant les principes des approches, les outils utilisés, ainsi que les réflexions et progressions avec comparaison des résultats obtenus pour chaque apport proposé. L'analyse devra comporter une évaluation des méthodes et chercher à identifier les cas dans lesquels l'algorithme est performant, et les cas dans lesquels l'algorithme est le moins bon.
  - Format attendu : PDF uniquement, 10 pages maximum (hors page de garde)
  - Informations élémentaires à ne surtout pas oublier : vos noms, date de réalisation, introduction et conclusion
  - Illustrations : citer la source quand elles ne sont pas de vous
  - Algos/codes : à éviter autant que possible, le rapport n'est pas une explication de code (et nous demandons l'accès au code par ailleurs). Si vous estimez néanmoins nécessaire d'en parler, écrivez-le sous forme de pseudo-code générique
  - Rigueur scientifique : vous devez justifier toutes vos affirmations par des faits clairement explicites, faute de quoi elles n'ont aucune valeur
  - Suggestion : vous pouvez utiliser le template IEEE : <https://www.ieee.org/conferences/publishing/templates.html>
- **Code source** : Chaque groupe doit fournir aussi son code source en python3, pour nous permettre de le vérifier et de le valider.
  - Format attendu : un lien vers son dépôt gitlab de l'UTC, à mettre dans le sous-groupe SY32/Panneaux P24/codes (<https://gitlab.utc.fr/sy32/panneaux-p24/codes>). Nous vous y avons ouvert un accès « Developer ». **Merci de nous signaler si vous n'y avez pas encore accès !**
  - requirements.txt : fournir un fichier requirements.txt compatible avec pip ou conda (liste les versions des librairies utilisées, pour une réinstallation facile)
  - Commentaires : commentez proprement vos sources
  - Chemins vers d'autres fichiers : proscrivez l'usage de chemins absolus, à moins de demander explicitement à l'utilisateur de les spécifier
  - Note : Les codes pourront éventuellement être utilisés ou servir de base pour le challenge UTAC. Dans ce projet, la rapidité d'exécution est aussi un critère important.

### **Métriques pour évaluer vos algorithmes**

- Matrice de confusion
- Précision et rappel
- mAP : mean Average Precision, moyenne des AP pour chaque classe

### **Barème**

Le rapport est noté sur 16 points :

- partie algorithme classique : 10 points
- partie apprentissage profond : 6 points

Les performances sont notées sur 4 points, **selon la métrique mAP** :

- 2 points pour le top 3 des meilleurs binômes dans la catégorie des algorithmes classiques
- 2 points pour le top 3 des meilleurs binômes dans la catégorie des algorithmes d'apprentissage profond
- de la même manière, 1 point pour le top 6 dans chaque catégorie