## Image parsing

Dans cet algorithme, l'objectif est de d'implémenter un algorithme qui permet de lire un fichier image en format brute et le décoder, en suite le compresser avec une compression sans perte.

Nous avons utilisé un fichier .bin qui est un enregistrement issu d'une caméra OpenMV CAM, dont la capture a été faite en utilisant un programme sur micro python.

J'ai écrit un code en micro python pour la lecture du fichier bin mais on n'a pas pu le tester malheureusement vu que je ne possède pas le matériel (Open MV CAM)

# Création d'un dataset pour la détection d'objet

En utilisant un ensemble d'images synthétiques générées à partir de la combinaison d'une image background et d'une image Object qui correspond à une bouteille. L'idée c'était de positionner la bouteille sur des positions assez différentes dans l'arrière-plan correspondant à l'image background. Comme le montre l'image suivante :

Dans l'image ci-dessus, on peut voir clairement l'image objet se positionner dans l'image background. L'idée est de créer une base de données permettant de créer, tester et valider un modèle de machine Learning YOLO.

A l'entrée, nous donnons un dossier contenant les images et à la sortie, nous aurons notre dataset qui contient un dossier pour les données de training, un dossier de données de test et un dossier de données de validations et chaque dossier contient un fichier .json basé sur le style COCO. Le style COCO est un fichier .json qui se caractérise comme le format ci-après.

```
"info"
{
"year": int,
 "version": str,
 "description": str,
 "contributor": str,
 "url": str,
 "date_created": datetime,
}
```

```
"info":
    {
        "year": 2019,
        "version": "1.0",
        "description": "Flower and Fruits dataset",
        "contributor": "Flowers Inc.",
        "url": "http://test.org",
        "date_created": "2019/12/04"
    }
```

```
Categories
[{
"id": int,
 "name": str,
 "supercategory": str,
}]
```

```
"categories":
[
    {"id": 1,"name": "rose", "supercategory": "flower"},
    {"id": 2,"name": "tulip", "supercategory": "flower"},
    {"id": 10,"name": "Apple", "supercategory": "fruit"}
]
```

```
annotation{
"id": int,
 "image_id": int,
 "category_id": int,
 "segmentation": RLE or [polygon],
 "area": float,
 "bbox": [x,y,width,height],
 "iscrowd": 0 or 1,
}
```

```
"annotations": [
    {
        "segmentation":
[[510.66,423.01,511.72,420.03,...,510.45,423.01]],
        "area": 702.10,
        "iscrowd": 0,
        "image_id":  397133,
        "bbox": [433.07,355.93,138.65,228.67],
        "category_id": 18,
        "id": 1768
    },
    {
        "segmentation":
      {
        "counts":[12,56,198,10]
        "size":[120, 240]
      }
        "area": 500.2,
        "iscrowd": 1,
        "image_id": 397122,
        "bbox": [473.07,395.93,38.65,28.67],
        "category_id": 18,
        "id": 1768
    }]
```

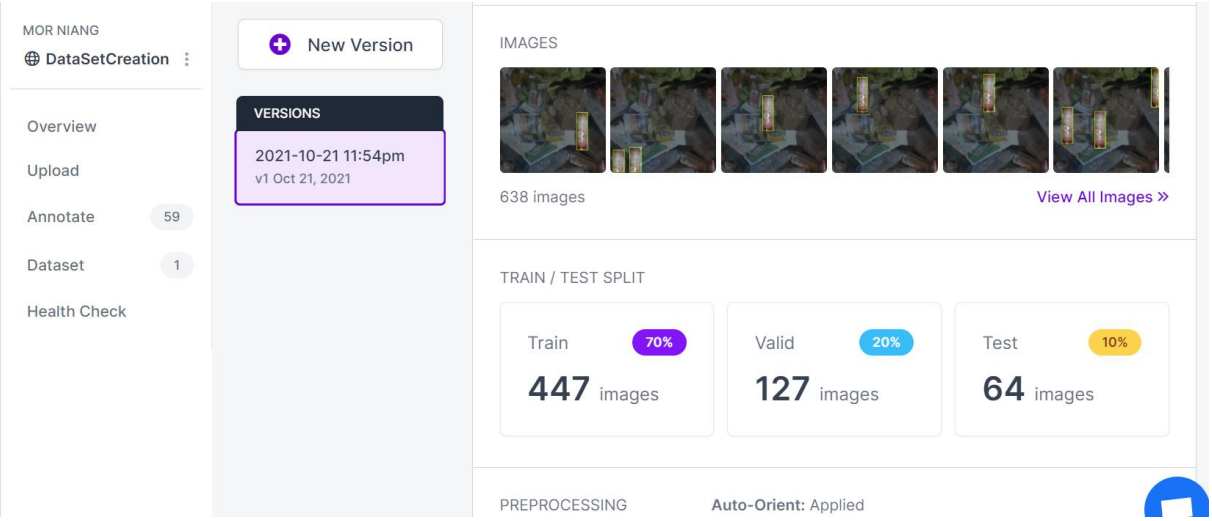
```
image{
"id": int,
 "width": int,
 "height": int,
 "file_name": str,
 "license": int,
 "flickr_url": str,
 "coco_url": str,
 "date_captured": datetime,
}
```

```
 "images": [
     {
         "id": 397133,
         "width": 640,
         "height": 640,
         "file_name": "101.jpg",
         "license":1,
          "date_captured": "2019-12-04 17:02:52"
     },
     {
          "id": 397122,
         "height": 640,
         "width": 640,
         "file_name": "102.jpg",
         "license": 1,
         "date_captured": "2019-12-04 17:02:52"
     }
     ]
```

Nous avons utilisé un outil qui s'appelle **Roboflow,**

On a commencé par importer l'ensemble du dossier qui contient les images, ensuite nous avons divisé les données en des données de training, de test et de validation et enfin, pour chaque image, encadrer l'élément objet qui correspond à la bouteille, cette étape, nous

spécifier le rectangle qui contient l'objet dans l'image background. Et après cet étape, nous avons générer notre Fichier **Json de format COCO.**




## Machine Learning

En machine Learning, comme tout apprentissage machine, le processus d'apprentissage est presque le même. On divise les données en trois parties (Données pour l'apprentissage, données pour la validation et des données pour le test). Pour faire un bon apprentissage, il faut prendre d'une manière aléatoire 70 pourcents des données pour le Training et le reste des données pour le test et validation. Normalement si on a des données cohérentes, on doit avoir des graphes qui se ressemblent entre le test et la validation. Un écart entre le résultat des tests et de validations, signifie qu'il y'a un overfit ou le split entre training, test et validation n'est pas correct.

## Libraries utilisés

### Opencv :

Module utilisé pour le traitement des images sous python, il contient plusieurs fonctions permettant de réaliser des opérations sur les images, comme le filtrage, les déformations etc.

Pour l'installer on utilise souvent la commande **pip install opencv-python**

### Matplotlib

Ce module est utilisé pour gérer les graphiques sur python, comme l'affiche des images et des graphes en général

### Pillow Image

Quant à ce module est utilisé souvent pour gérer les entrées sorties des fichiers images, comme la lecture, l'écriture

## Algorithme d’Image Processing Salades