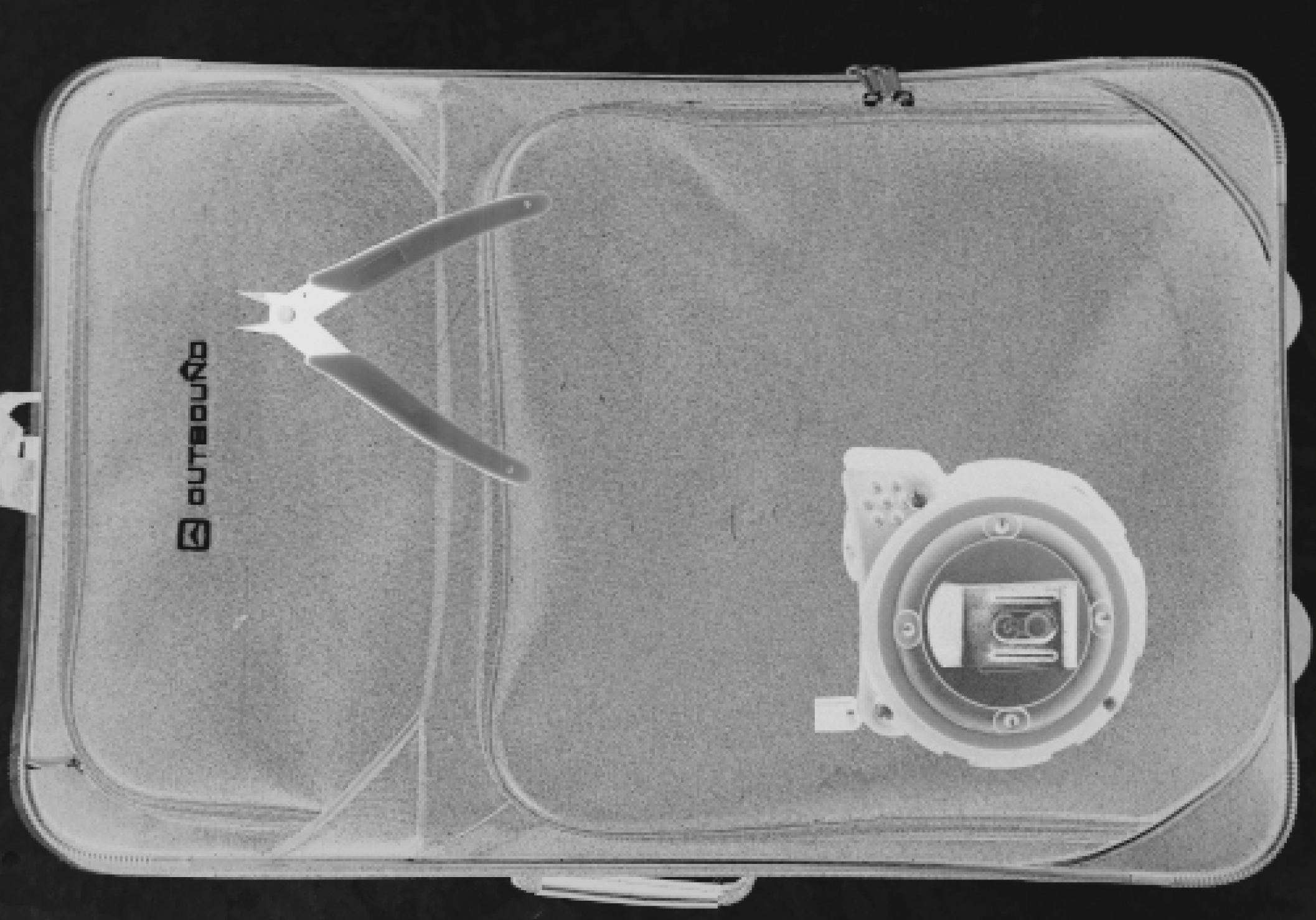
Problématique

Le but du projet est de simuler, à l’aide de réseaux de neurones convolutifs, un système d’examination par balayage (scanner, comme celui utilisé pour le scan des bagages.



Nos réseaux devront être capable de classifier, détecter et segmenter différentes formes dans une image. Pour se faire nous utiliserons la librairie PyTorch concevoir et entrainer le réseau. Afin de vérifier l’implémentabilité de chacune des techniques, il nous est demandé de fournir une architecture indépendante pour chacune des tâches. Un dataset comportant les différentes nous est fourni avec les caractéristiques suivantes :

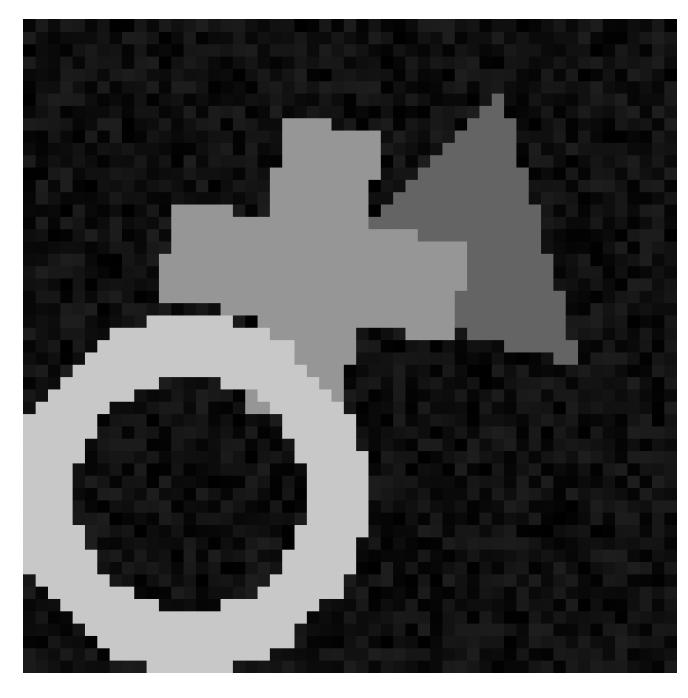
• Un maximum de trois formes par images;

• Un maximum d’une instance de chaque forme par images;

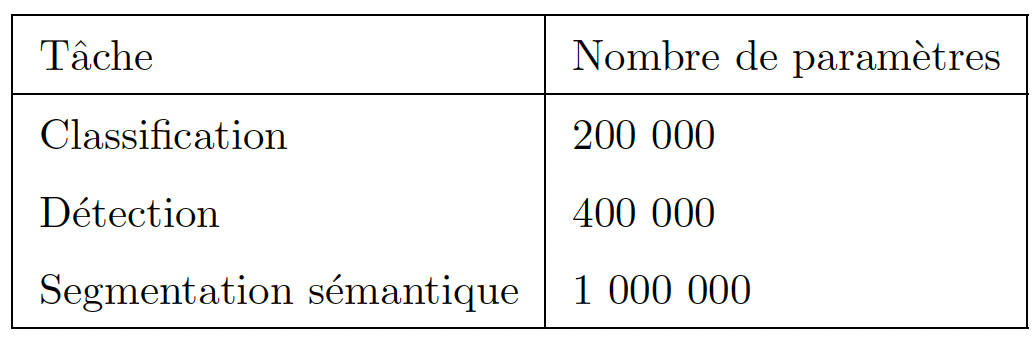
• Images de dimensions 53 pixels par 53 pixels, en niveaux de gris;

• Le niveau de gris est aléatoire et indépendant de chaque forme;

• L’arrière-plan est bruitée et tend vers le noir.



De plus nous sommes limités au niveau de la mémoire par un nombre de paramètres maximum pour chaque tâches.



Une description des modules pour charger les données d’entraînement et de validation

\_\_len\_\_()

Retourne la taille du data set, qui est égale au nombre de clé stocké dans le dictionnaire \_metadata.

\_\_getitem\_\_(index)

Retourne image, segmentation\_target, boxes, class\_labels tel que

Image : Est un tenseur de taille 1x53x53 contenant les niveaux de gris de l’image index.

segmentation\_target : Est un tenseur de taille 53x53 où chaque valeur correspond à la classe active du pixel.

boxes : Est un tenseur de taille 3x5 tel que

(i,0) sont à 1 lorsque la bounding Box est active et à 0 si elle est inactive.

(i,1) est la position x en % de l’image du centre de la bounding box.

(i,2) est la position y en % de l’image du centre de la bounding box.

(i,3) est la taille de la bounding box où sa hauteur est égale à sa largeur.

(i,4) correspond à la classe active de l’objet

class\_labels : est un tenseur de taille 3 tel que la valeur à l’indice i est mis à 1 si la classe i est active et à 0 si elle est inactive.

I=0 cercle

I=1 triangle

I=2 croix

Implémentation de la classe ConveyorSimulator :

La classe hérite de la classe torch.utils.data.Dataset. Elle permet de stocker les images et leurs labels correspondant. Pour ce faire nous stockons dans un dictionnaire \_metadata les différentes images. Les données associées aux images sont elles aussi stockées dans un dictionnaire contenant les clés position, shape et size.

La fonction \_\_len\_\_ permet d’avoir le nombre d’images stockées dans \_metadata.

La fonction \_\_getitem\_\_ permet d’avoir les données relatives à l’image i.

Cette classe est utilisée pour stocker les dataset d’entrainements, de validation et de test.

torch.utils.data.DataLoader

batch\_size définit la taille par laquelle on sépare les données en sous ensemble que nous ferons passer dans le réseaux.

Mettre le shuffle à true permet de mélanger les données à chaque époque, cela permet de réduire le sur-apprentissage du modèle.

num\_workers permet de spécifier le nombre de coeurs utilisé par le processus.