Documentation technique

Projet 1.02 : Tri de déchets (projettridedechets@gmail.com)

Amine CHERIF HAOUAT
Tom LABIAUSSE
Cyrine NABI
Pierre OLLIVIER
Selim BEN TURKIA

[0] Sommaire

- [1] Description du projet
- [2] Vue d'ensemble de l'architecture
- [3] Architecture détaillée des entités et procédures
  - [3.1] Entités de la structure locale du projet
  - [3.2] Procédure de téléchargement des données de TACO
  - [3.3] Procédure de normalisation de la base de données TACO
  - [3.4] Structure de la base SQL contenant les métadonnées
  - [3.5] Procédure d'jout d'une nouvelle catégorie
  - [3.6] Procédure de fusion avec la base de données TIDY
  - [3.7] Procédure d'ajout de nouvelles images
  - [3.8] Redimensionnement des photos pour les CNN
  - [3.9] Entraînements des CNN
  - [3.10] Utilisation du système final de classification

[1] Description du projet

L'objectif du projet *Tri de déchets* est de mettre au point un algorithme d'intelligence artificielle permettant de classifier des déchets abandonnés dans l'espace publique à partir de photographies de ces derniers. Ces classes sont relatives à la nature du matériau (plastique, carton, verre…) constituant le déchet mais aussi à sa forme (bouteille, canette, sac…). Le projet répond à une demande de l'association *Eaux Cristallines* qui souhaiterait pouvoir automatiser la détection et la classification de déchets à partir de photos afin d'avoir une meilleure vision de la répartition de ces derniers dans la nature. A terme, cette cartographie des déchets devrait permettre de mener des actions ciblées de rammassage pour lutter contre cette pollution nottament en bord de mer.

Ce projet utilise des méthodes de Deep Learning pour la classification et se découpe en deux parties principales :

- **1)** Constitution et gestion d'une base de données de photos couleurs de déchets abandonnés avec leur labelisation.

- **2)** Construction et entrainement de plusieurs réseaux de neurones convolutifs (CNN) pour la classification de déchets à partir de photos.

Les programmes présentés dans toute cette documentation sont écrits en langage Python ou SQL. Le code informatique Python de la partie **i)** est compatible avec les versions 2.7.15 et 3.8.3 tandis que le code de la partie **ii)** n'est supporté que par la 3.8.3 car il fait appel à la bibliothèque plutôt récente de Machine Learning appelée *TensorFlow*.

Ce projet utilise principalement la base de données TACO *(Trash Annotations in COntext)* regroupant 1500 photos couleurs dans des formats de l'ordre de 3200x2400 pixels. Cette base de donnée est accessible librement à l'adresse : https://github.com/pedropro/TACO. Le projet exploite également la base de données TIDY *(Trash Image Dataset 'Yucky')* également accessible librement à l'adresse https://github.com/gale31/TIDY.

[2] Vue d'ensemble de l'architecture

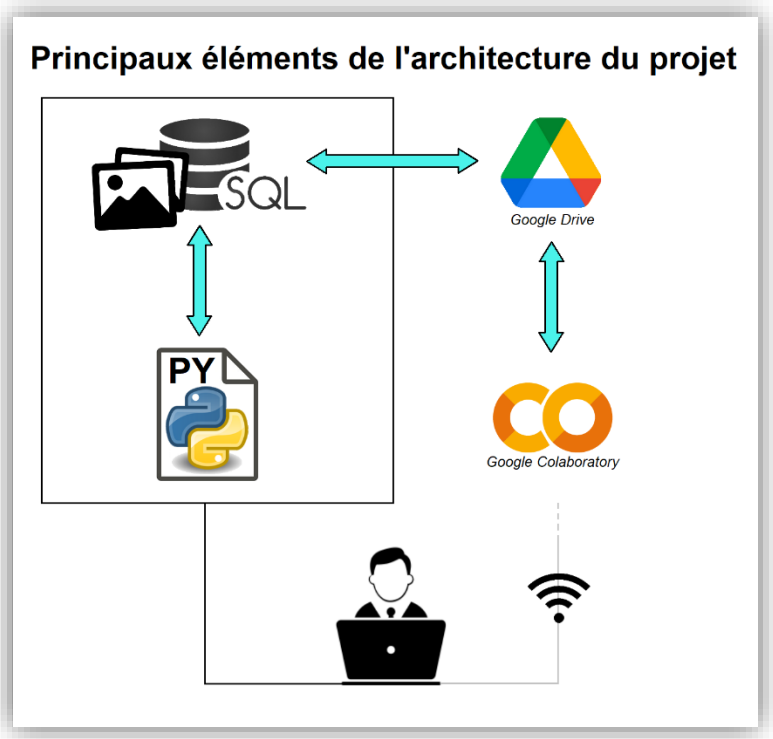

Figure 1 : Principaux éléments de l'architecture du projet

L'architecture des entités du projet peut se découper en deux parties :

- D'une part, la gestion de la base de données est réalisé en local sur l'ordinateur de l'utilisateur. Cela requiert d'utiliser des scripts Python ainsi qu'une base SQL pour stocker toutes métadonnées relatives aux photos.

- D'autre part, pour des raisons de temps de calcul, l'entrainement des CNN est difficilement réalisable en n'exploitant seulement que les ressources de l'ordinateur de l'utilisateur. Ainsi, il est nécessaire d'avoir recours à un environnement de travail en ligne tel que *Google Colaboratory*. Cependant, toutes les ressources utilisées dans cet environnement doivent y être

préalablement chargées. De plus, la mémoire de cet environnement est réinitialisée à chaque nouvelle connexion. Ainsi, il n'est pas raisonnable de vouloir charger toutes les données nécessaires à l'entrainement des CNN à chaque séance de travail. Pour remédier à cela, une fois la base de données organisée et préparée en local, nous avons décidé de charger cette dernière dans un *Google Drive* à l'adresse mail du projet projettridedechets@gmail.com. Une fois ce chargement réalisé, il est ensuite possible d'accéder rapidement aux données stockées sur le *Google Drive* via *Google Colaboratory*.

Au cours du projet, nous avons été amené à enrichir la base de données initiales. Cependant, le système que nous avons mis en place ne permet pas de modifier directement la base de donnée stockée sur *Google Drive*. Ainsi, il est nécessaire de modifier la base de donnée en local dans un premier temps avant de mettre à jour celle disponible sur *Google Drive*. Cette procédure pourrait poser problème si on était amener à gérer des données de taille beaucoup plus importantes. Cela n'a pas été le cas dans ce projet car la base de données stockée en ligne était de l'ordre de 100Mo.

Les entités de la partie locale du projet peuvent être obtenues en téléchargeant le fichier *« Projet_TDD_LOCAL.zip ».* Le contenu du fichier dezipé est représenté ci-dessous :

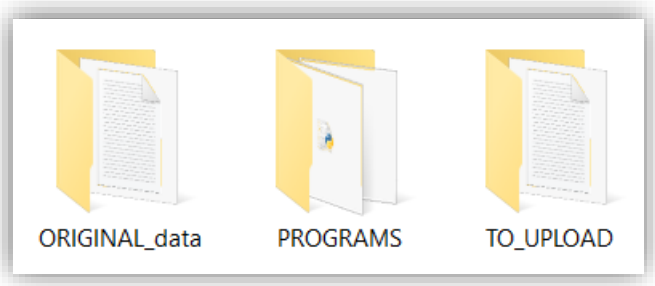

Figure 2 : Contenu initial du dossier Projet_TDD_LOCAL

Le nom du dossier *Projet_TDD_LOCAL* peut être modifié mais pas celui des trois dossiers de la *Figure 2* qui doivent rester ensemble dans un même emplacement au sein de l'ordinateur de l'utilisateur.

La partie consacrée au Deep Learning du projet est constituée du notebook *Projet_TDD_CNN.ipynb* exécutable sous *Google Colaboratory*.

[3] Architecture détaillée des entités

[3.1] Entités initiales de la structure locale du projet

On détaille dans cette partie le contenu et le rôle des entités de *Projet_TDD_LOCAL* dont un aperçu est donné en *Figure 2*.

- ***ORIGINAL_data*** **:** Dossier contenant la base de données TACO originale. Les métadonnées sont stockées dans le fichier *metadata_TACO.txt*

- ***TO_UPLOAD*** **:** Dossier servant de zone de transit pour des images supplémentaires à charger dans la base de données. Contient un fichier *new_labels_LIST.txt* servant à labelliser manuellement les nouvelles images

- ***PROGRAMS*** **:** Dossier contenant 7 scripts python ainsi qu'un dossier *ANNEXES* pouvant contenir d'éventuelles ressources pour des interfaces graphiques par exemple.

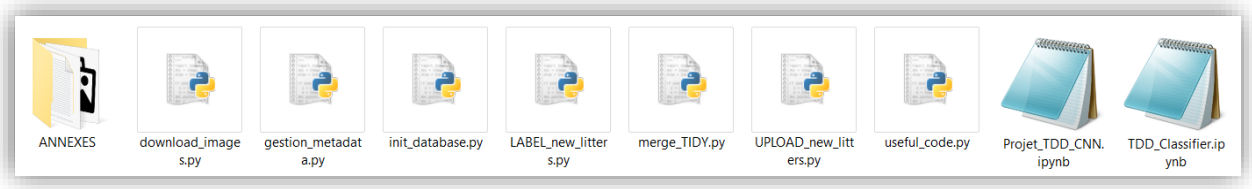

Figure 3 : Contenu du dossier PROGRAMS

- ***download_images.py*** **:** Script de téléchargement des images de la base de données TACO.
- ***init_database.py*** **:** Script de normalisation des données et de construction de la database SQL rassemblant les métadonnées associées aux photos.
- ***gestion_metadata.py*** **:** Script rassemblant diverses fonctionnalités implémentées sur la base de données (affichage de statistiques, visualisation d'une image, modification de données…).
- ***merge_TIDY.py :*** Script permettant de fusionner la base de données TACO avec les images de la base de données issue de TIDY.
- ***LABEL_new_litters.py :*** Script de l'interface graphique permettant de labelliser de nouvelles images présentes dans TO_UPLOAD.
- ***UPLOAD_new_litter.py :*** Script permettant de charger dans la base de données des images préalablement labellisées manuellement où à l'aide de LABEL_new_litters.py.
- ***useful_code.py :*** Script rassemblant diverses fonctions employées dans les script précédents mais ne réalisant pas des taches de grand intérêt à elles seules.
- ***Projet_TDD_CNN.ipynb :*** Script permettant d'entrainer les CNN sur la base de donnée construite et de sauvegarder les modèles obtenus.
- ***TDD_Classifier.ipynb :*** Script permettant à partir des modèles sauvegardés de réaliser des classification de déchets sur de nouvelles images.

[3.2] Procédure de téléchargement des images de TACO

Télécharger les images de la base de données TACO nécessite une connexion Internet ainsi que le script *download_images.py*. Il suffit d'exécuter ce script et toutes les images seront téléchargées dans le dossier *ORIGINAL_data*. Une fois le téléchargement terminé, ce dossier doit contenir le fichier *metadata_TACO.txt* ainsi que 15 dossiers nommés *batch_x* pour x allant de 1 à 15.

[3.3] Procédure de normalisation de la base de données TACO

Utiliser des photos pour entrainer des CNN nécessite d'avoir un format unifié en entrée ainsi qu'un moyen rapide d'accéder aux métadonnées. Ce n'est pa le cas de la base de données TACO telle qu'elle a été téléchargé précedemment. La procédure de normalisation permet de rémédier à cela en créeant une copie analysée de la base de données qui sera stockée dans un dossier nommé *RESIZED_data*. Réaliser cette normalisation nécessite d'avoir déjà réalisé l'étape [3.2] de téléchargement puis d'exécuter le script *init_database.py*.

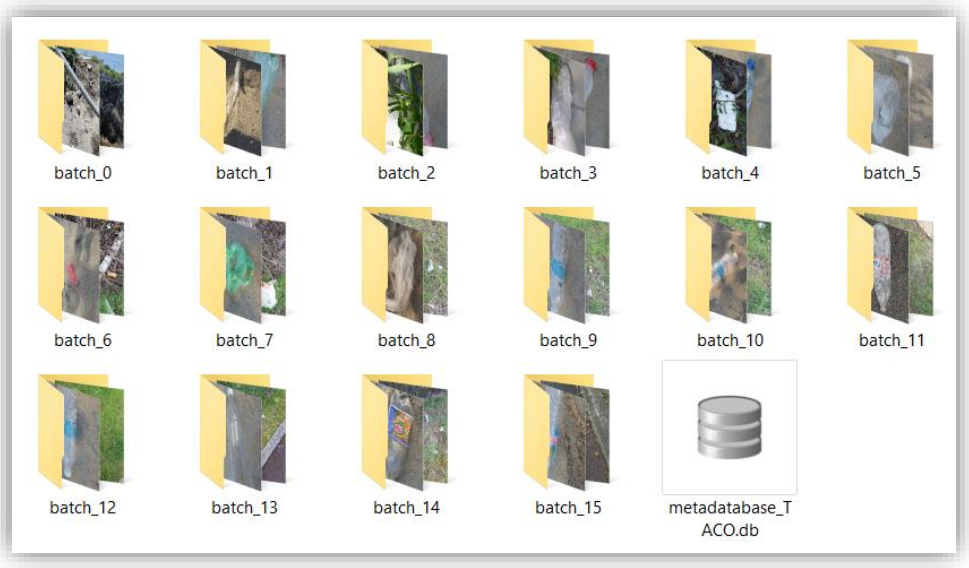

Figure 4 : Contenu du dossier RESIZED_data

La *Figure 4* donne un aperçu du contenu de *RESIZED_data* après normalisation de la base de données. Les métadonnées sont réunies dans la base SQL *metadatabase_TACO.db*. Nous avons fait le choix de mettre à l'écart les photos dont le format est sensiblement différent du format 4/3 (format majoritaire dans la base de données TACO). Ainsi, toutes les images ne pouvant pas être redimensionnées au format 4/3 sans subir de déformation trop importante sont stockées dans le dossier *batch_0*. Les *batch_x* pour x allant de 1 à 15 présents dans *RESIZED_data* contiennent au final uniquement des images au format 3264/2448 (= 4/3) en mode portrait. Ce format 4/3 particulier a été choisi car il correspond actuellement à la taille la plus courante des photos prises par des smartphones et est en effet le format 4/3 majoritaire dans la base de données TACO.

[3.4] Structure de la base SQL contenant les métadonnées

La base SQL contenant les métadonnées que nous avons décidé de conserver est enregistrée dans le fichier *metadatabase_TACO.db*. La *Figure 5* donne la structure de cette base de données.

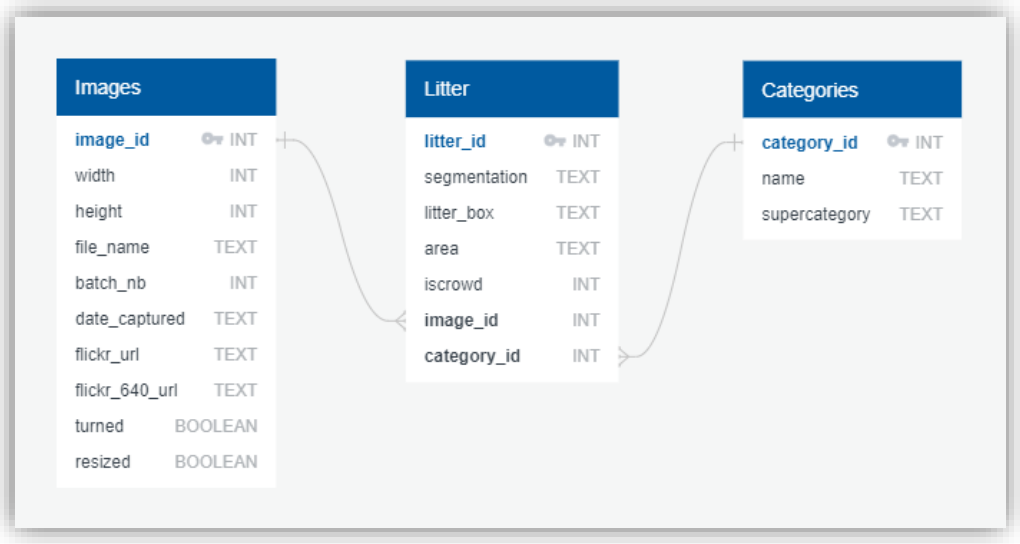

Figure 5 : Structure de la base SQL des métadonnées

Chaque image est repéré par un identifiant (*image_id*). On renseigne également les dimensions originales en pixels de la photo (*width,height*) avant toute transformation ainsi que le numéro de batch dans laquelle elle se situe au sein de *RESIZED_data*. Le champ *file_name* donne le chemin d'accès direct à la photo depuis l'intérieur de *RESIZED_data*. La date à laquelle a été prise la photo peut être renseignée dans *data_captured*. Les champs *flick_url* et *flick_640_url* sont hérités de la base de données TACO et donne una accès direct à la photo stockée en ligne. Enfin, *turned* et *resized* indiquent si lors de la normalisation décrite en [3.3] la photo a été tourné de 90° dans le sens trigonométrique et/ou redimensionné d'un format 4/3 quelconque au format 3264/2448.

Chaque déchet est repéré par un identifiant (*litter_id*). Les champs *segmentation* et *bbox* donnent des listes de coordonnées utiles pour localiser et détourer un déchet sur une photo et *area* donne la surface délimitée par la segmentation. *is_crowd* indique si un déchet sur une photo est scindé en plusieurs parties ou non.

Chaque catégorie de déchet est repérée par un identifiant (*category_id*). Les différentes catégories de la base de données TACO sont regroupées en 27 grandes familles appelées *supercategory* telles que (Bottle,Can,Cup,…). Au sein de chaque catégorie, on distingue plusieurs sous-catégories apparaissant via le champ *name* dans la table *Categories*. Par exemple, la *supercategory* Can se divise en trois sous-catégories Food Can, Drink can et Aerosol. A chaque déchet est alors associé une *category_id* correspondant à une sous-categorie *name.*

Cependant, la base de données TACO rassemble 60 sous-catégories. Or, ce nombre est beaucoup trop élevé car même si on supposait la base de données idéalement remplie, on disposerait de 1500/60=25 photos par sous-catégorie et donc

bien trop peu pour entrainer un CNN à reconnaître une sous-catégorie de déchet. Pour cette raison, nous avons décidé de nous focaliser uniquement sur les supercatégories dont le nombre de déchets nous paraissait suffisant et produisait d'assez bons résultats. Nous n'avons pas pour autant supprimer le champ *name* de notre base de données par mesure de précaution si d'aventure nous lui trouvions une utilité. Comme décrit en [3.7], nous avons décidé de labelliser les nouveaux déchets uniquement en leur associant une *supercategory* et non une sous-catégorie *name*. Pour rendre cela compatible avec la base de données, nous avons décidé de créer pour chaque *supercategory* la sous categorie *name=supercategory*. Cette catégorie « double » permet alors de labelliser les déchets par sous-categorie et l'opération de création de ces doubles catégories est réalisée automatiquement lors de l'étape de normalisation [3.3] grâce à la fonction *create_double_cat* issue du script *gestion_metadata.py*.

[3.5] Procédure d'ajout d'une nouvelle catégorie

Il peut être nécessaire d'ajouter une nouvelle catégorie de déchets à la base SQL afin de référencer de nouveaux déchets sur de nouvelles photos. Pour cela, il convient d'utiliser le script *gestion_metadata.py* contenant la fonction *add_new_cat* qui réalise cette tache.

[3.6] Procédure de fusion avec la base de données TIDY

Les images de la base de données TIDY ainsi que leurs métadonnées peuvent être ajoutées à la base de donnée construite à l'étape [3.3]. Pour ce faire, il convient d'utiliser le script *merge_TIDY.py*. Les instructions relatives à cette procédure sont directement intégrées dans ce script. Notons seulement que les catégories définies dans cette base de donnée ne correspondant pas exactement aux notres, un traitement manuel de sélection des données est difficilement évitable. La base de données TIDY ne rassemblant qu'environs 250 photos, cela a pu être réalisé.

[3.7] Procédure d'ajout de nouvelles images (illustrée avec MASK)

Il n'est possible d'ajouter une image à la base de données seulement si celle-ci est au format 4/3 au 1/1. L'ajout d'une image nécessite de lui associer un *image_id* ainsi qu'un numéro de batch de destination puis de labelliser tous ses déchets et de tous leur associer un *litter_id*. On fait alors référence à tout cela par l'étape d'étiquetage. Cet étiquetage peut être réalisé manuellement ou être facilité grâce à une interface graphique.

1) Etiquetage facilité : Placer l'image à étiqueter dans le dossier *TO_UPLOAD* puis lancer le script *LABEL_new_litters.py* permettant d'ouvrir une interface graphique.

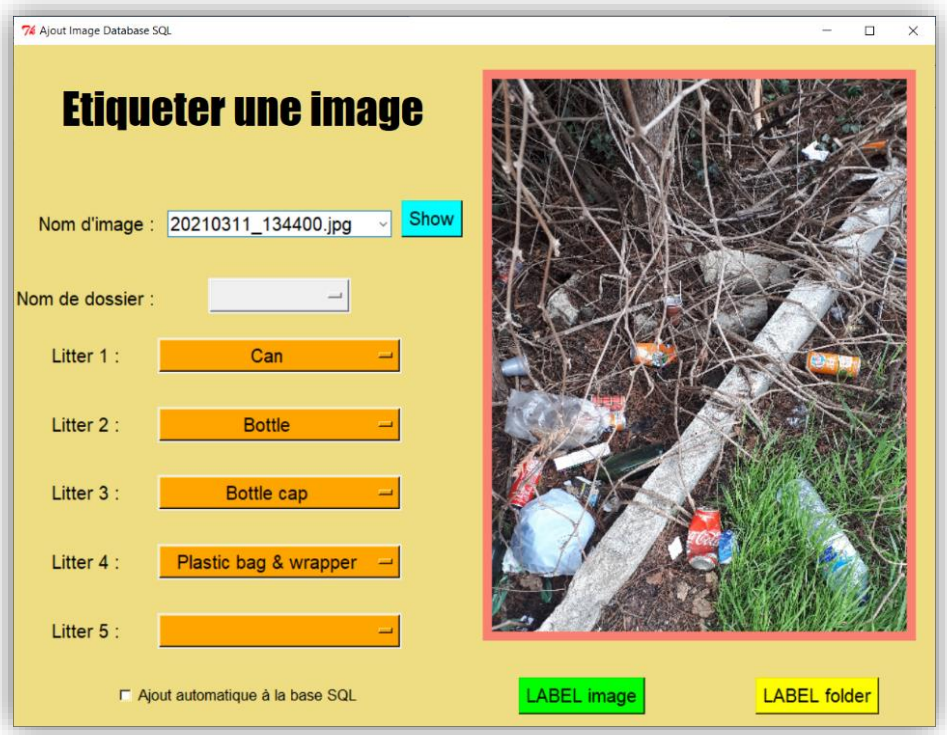

Figure 6 : Exemple d'utilisation de la fenêtre d'étiquetage facilité

Il est alors possible de sélectionner une image à étiqueter grâce au champ *Nom d'image*. Il suffit ensuite de renseigner les différentes catégories de déchets présents sur la photo. Ces catégories correspondent exactement aux *supercategory* de la base de données SQL. Ainsi, dans l'exemple de la *Figure 5*, l'image sera déclaré avec 4 types de déchets dont les couples *(supercategory,name)* sont les suivants : *(Can,Can), (Bottle,Bottle), (Bottle cap,Bottle cap), (Plastic bag & wrapper, Plastic bag & wrapper).* Une fois ces informations renseignées, il suffit de valider grâce au bouton *LABEL image.* Les informations renseignées sont alors directement écrites dans le fichier *new_labels_LIST.txt* dont un aperçu est donné en *Figure 6.*

Il est également possible d'étiqueter en une seule fois plusieurs images. Cela peut également être utilisé lorsque l'utilisateur dispose de photos contenant toutes les mêmes types de déchets. Pour ce faire, il suffit de placer toutes ces photos dans un même dossier lui-même positionné dans le dossier *TO_UPLOAD.* Il faut ensuite ouvrir l'interface graphique, sélectionner le dossier contenant les photos grâce au champ *Nom de dossier*, renseigner les supercatégories de déchets puis valider avec le bouton

LABEL_folder. Comme dans le cas de la photo unique, toutes les informations sont directement renseignées dans le fichier *new_labels_LIST.txt*.

Pour ne pas trop alourdir l'interface, il n'est possible d'ajouter que 5 types de déchets par photo au maximum. Cependant, si une photo contient au moins 6 types différents de déchets, ces derniers peuvent être référencés grâce à l' étiquetage manuel décrit dans le paragraphe suivant.

2) Etiquetage manuel : Placer tout d'abord l'image à ajouter dans le dossier *TO_UPLOAD.* Le fichier *new_labels_LIST.txt* est organisé en lignes correspondant chacune à un déchet sur une image. L'entête ne doit pas être modifiée et donne la nature des 5 informations nécessaire à chaque ligne. Le séparateur utilisé sur une ligne est « ---- ».

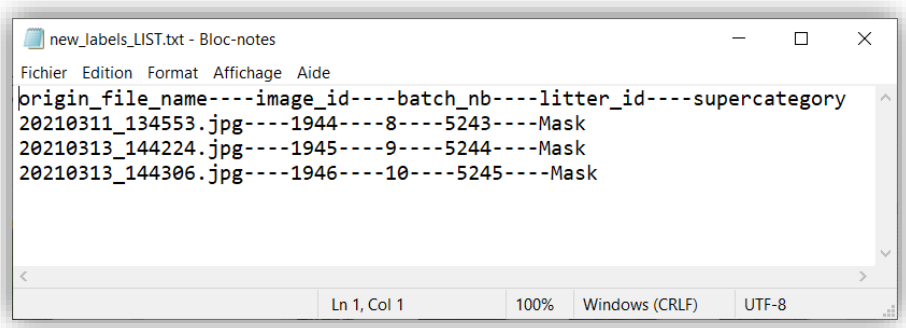

Figure 7 : Exemple d'utilisation du fichier new_labels_LIST.txt

Une fois l'étape d'étiquetage terminée, tout est alors en place pour ajouter le/les image(s) à la base de données ainsi que les métadonnées. Il suffit alors d'exécuter le script *UPLOAD_new_litters.py*. Ce dernier permet, en sélectionnant « non » pour l'ajout réel des images, de vérifier la cohérence des informations du fichier *new_labels_LIST.txt* avec la base de donnée existante. C'est une sécurité permettant d'identifier d'éventuelles erreurs pouvant êtres commises lors de l'etiquetage manuel. En selectionnant « oui » pour l'ajout réel des images, les photos dont des déchets ont été etiqueté sont copiées dans la base de donnée *RESIZED_data* et les métadonnées de *new_labels_LIST.txt* ajoutées à la base de donnée SQL *metadatabase_TACO.db* présent dans *RESIZED_data*.

Il est possible de réaliser l'étape d'ajout à la base de données automatiquement lors de l'étape d'étiquetage en cochant la case « Ajout automatique à la base SQL » dans l'interface graphique.

[3.8] Redimensionnement des photos pour les CNN

Les CNN utilisés pour la classification binaire de déchets nécessitent des images au format carré en entrée. Or, jusque là nous avons toujours privilégié les formats 4/3 car c'est le format de photos le plus courant au sein de la base de données TACO. Cependant, nous avons estimé que la déformation d'une photo ocasionnée en passant son format de 4/3 à carré n'était pas « trop importante » dans le sens où un observateur humain parvient à y reconnaître aisément l'objet d'origine représenté. De plus, pour certaines photos, il n'est même pas évident qu'un observateur humain

parvienne à voir que leurs formats ont été modifiés. Pour ces raisons, nous avons décidé de passer toutes les photos de notre base de données au format carré pour entrainer les CNN. Pour ce faire, il suffit d'utiliser la fonction *copy_resize* présente dans le script *gestion_metadata.py.* Cette fonction prend en paramètres deux entiers *new_width* et *new_height* qui sont les nouvelles dimensions souhaitées pour les photos. L'exécution de *copy_resize* créé un nouveau dossier *RESIZED_data_nwxnh* (*nw* et *nh* étant les deux nouvelles dimensions) et y copie la totalité des batch 1 à 15 présents dans *RESIZED_data* en mettant les photos au format désiré. Cette copie peut sembler lourde mais le modèle d'architecture de CNN utilisé étant celui de VGG-16, nous avons du redimensionner les photos en 244x224 pixels ce qui réduit considérablement le poids des données et nous amène à ne copier qu'aux alentours de 60Mo de données, cela reste raisonnable.

Comme expliqué lors des étapes [3.6] et [3.7], la base de données construite est amené à être enrichie. Ces mises à jour de la base de données n'ont cependant d'effet que sur le dossier *RESIZED_data*. Pour également mettre à jour notre dossier *RESIZED_data_224x224*, nous avons implémenté la fonction *update_copy_database* disponible dans le script *gestion_metadata.py*. Cette fonction identifie les éléments manquants dans *RESIZED_data_224x224* par rapport à *RESIZED_data* et les ajoute au bon format.

Afin de finalement pouvoir entrainer nos réseaux de neurones, il convient comme expliqué en [2] de charger la totalité du fichier *RESIZED_data_224x224* sur un *Google Drive* depuis lequel les photos seront accessibles pour le script *Projet_TDD_CNN.ipynb* exécuté depuis *Google Colaboratory.* Il faut aussi copier le fichier *metadatabase_TACO.db* dans le dossier *RESIZED_data_224x224* présent sur *Google Drive* pour pouvoir également accéder aux métadonnées.

A ce stade et en ayant suivi toutes les procédures précédentes, l'utilisateur devrait être en possession d'un espace de travail comme celui-ci-dessous :

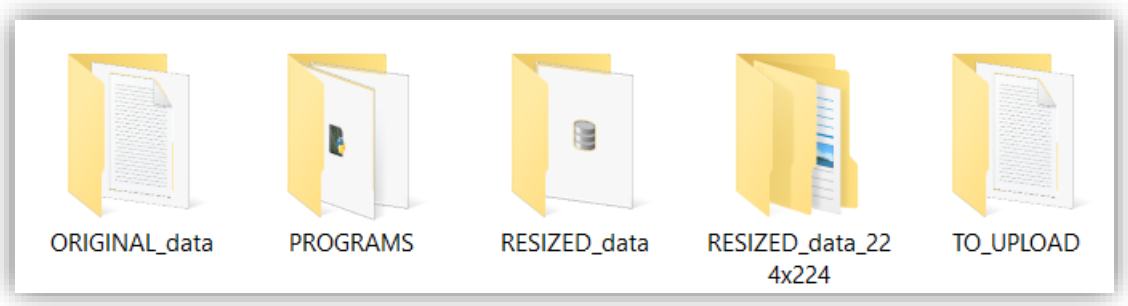

Figure 8 : Contenu du dossier Projet_TDD_LOCAL en mode fonctionel

Il est alors possible de supprimer le fichier *ORIGINAL_data* qui n'est désormais plus utile. Il pourrait même sembler bizarre de ne pas directement agir sur les données de *RESIZED_data_224x224* pour ainsi supprimer *RESIZED_data*. Nous avons décidé de ne pas faire cela car notre mode de fonctionnement permet de pouvoir produire à tout moment une copie de notre base de donnée avec les photos dans des dimensions quelconques à partir de *RESIZED_data* qui contient les photos originales et donc non déformées.

[3.9] Entraînement des CNN

Le script *Projet_TDD_CNN.ipynb* présent dans le dossier PROGRAMS contient tout le code nécessaire à l'entraînement des CNN. Il suffit alors d'ouvrir ce script dans *Google Colaboratory* et de suivre les instructions directement intégrées dans le code. Ce script permet d'entrainer et de sauvegarder directement dans *Google Drive* les modèles de classifieurs binaires pour différentes catégories de déchets.

[3.10] Utilisation du système final de classification

L'algorithme de classification permettant d'identifier les catégories de déchets présents sur une photo se trouve dans le script *TDD_Classifier.ipynb*. Pour fonctionner, l'algorithme doit pouvoir accéder aux modèles de classifieurs binaires enregistrés dans *Google Drive* lors de l'étape [3.9]. Une fois ces modèles à disposition, il suffit de suivre les instructions du script *TDD_Classifier.ipynb*.