



Detection automatique de pigments

Manuel Utilisateur

Version 1.0

Barrey Victor, Dupouey Léo, Moze Jonathan
April 22, 2023

Table des matières

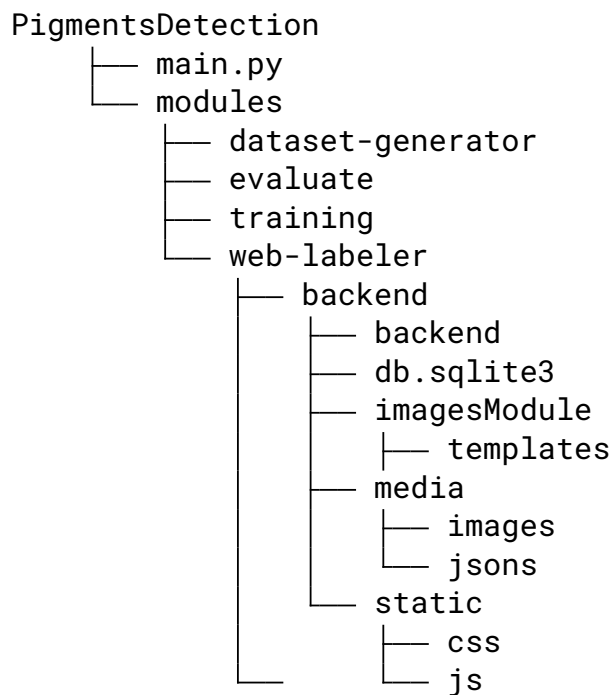
1	Introduction	2
2	Architecture de l'application	2
3	Guide d'installation	3
4	Utilisation de l'application	4
4.1	Etiqueteur web	4
4.2	Générateur du dataset	5
4.3	Entrainement du modèle d'apprentissage automatique	6
4.4	Evaluation d'un ensemble d'images	7

1 Introduction

Ce document est le manuel utilisateur de l'application de détection de pigments par apprentissage automatique. Il est composé de trois modules. Le premier est une application web permettant d'étiqueter les pigments d'une image, le second est un module permettant de générer le set d'apprentissage depuis les fichiers jsons d'étiquetage des images. Enfin, le dernier module est le module d'apprentissage automatique par réseau de neurones.

2 Architecture de l'application

Voici l'architecture générale de l'application, Nous y ferons référence plus tard dans le manuel.



3 Guide d'installation

Cette application est développée en **Python** sous la version 3.10.11, Il est donc recommandé d'utiliser cette version pour le garantir son bon fonctionnement. Python 3.10 est disponible à cette adresse :

<https://www.python.org/downloads/release/python-31011/>

Une fois python installé, nous allons créer un environnement virtuel python afin d'éviter les conflits avec les librairies que nous allons installer plus tard. Pour créer un environnement, utilisez la commande :

```
> python -m venv <environment name>
```

Ensuite, pour activer l'environnement virtuel, exécutez :

```
> source env/bin/activate
```

(sous windows exécutez) :

```
> env/Scripts/activate
```

À ce stade, votre terminal (selon celui que vous utilisez) ajoutera probablement le nom de votre environnement au début de chaque ligne de votre terminal (le nom donné à la place de **<environment name>**).

Nous allons maintenant installer les librairies dont nous avons besoin. Pour cela placez vous dans le dossier **PigmentsDetection**. Une fois cela fait, exécutez les commandes :

```
> pip install -r requirementsTorch.txt
```

```
> pip install -r requirements.txt
```

ces commandes installent toutes les librairies contenues dans les deux fichiers texte. Il est très important de les effectuer dans le bon ordre pour garantir une bonne installation de toutes les librairies.

Nous sommes maintenant en capacité d'exécuter notre application.

4 Utilisation de l'application

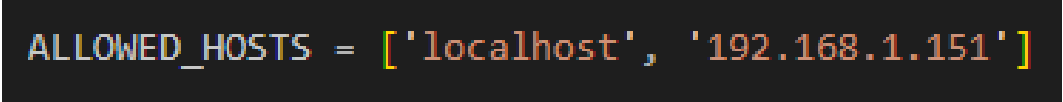
L'application fonctionne en ligne de commande, il suffit simplement de se place dans le dossier racine **PigmentsDetection**.

4.1 Etiqueteur web

Avant de lancer l'application web, si vous souhaitez que le serveur soit accessible sur un réseau local, il faudra au préalable récupérer l'adresse IPV4 locale de la machine où le logiciel est installé. Pour cela, sous linux tapez la commande **ifconfig** et l'adresse se trouve à la balise **inet**. Sous windows, tapez **ipconfig** dans un invite de commande et l'adresse se trouve à la balise **Adresse IPv4**.

Une fois l'adresse IPV4 récupérée, nous allons devoir la placer dans la balise **ALLOWED_HOSTS** du fichier **settings.py** dont le chemin relatif est :

```
> /modules/web-labeler/backend/backend/settings.py
```



```
ALLOWED_HOSTS = ['localhost', '192.168.1.151']
```

Figure 1: Exemple de la balise remplie avec une adresse IPV4

le serveur est maintenant prêt à être démarré. depuis le dossier racine, exécutez la commande suivante :

```
> python main.py runserver
```

Pour accéder au serveur depuis la machine ou il est lancé, allez sur un navigateur web et entrez l'url "<http://localhost:8000>". Si vous voulez y accéder depuis une autre machine en réseau local, entrez l'url "<http://<ip>:8000>" avec <ip> qui est l'adresse IPV4 que nous avons récupéré précédemment.

L'étiquetage peut ensuite être effectué. Les fichiers jsons créés sont stockés dans le dossier **jsons** dont le chemin relatif est :

```
> /modules/web-labeler/backend/media/jsons
```

4.2 Générateur du dataset

Une fois que les jsons sont récupérés, il faut maintenant générer le set d'entraînement.

tapez la commande :

```
> python main.py generate-dataset
```

avec les arguments :

`-i INPUT_DIR`

Path to the directory containing
'images\' and 'jsons\' subfolders.

`-o OUTPUT_DIR`

Path to the directory where the new
dataset will be written.

(OPTIONAL) `-s SUB_IMAGE_SIZE`

Size of the images to be created
for the dataset (Default 256).

(OPTIONAL) `-pp POSITIVE_PERCENTAGE`

Percentage of positive images to include
in the generated dataset (Default 100%)

(OPTIONAL) `-b`

If set, mask will be binary (black/white),
else each instance instance will be
encoded with a different shade of grey.

(OPTIONAL) `-h`

show the help message.

Une fois terminé, les patchs se trouvent dans le dossier **batch** dont le chemin relatif est :

```
> OUTPUT_DIR/batch
```

et les masques associés dans le dossier **mask** dont le chemin relatif est :

```
> OUTPUT_DIR/mask
```

Ce que contiennent ces deux dossiers forment le dataset d'entraînement dont nous aurons besoin pour entraîner ensuite le modèle.

4.3 Entraînement du modèle d'apprentissage automatique

pour lancer l'entraînement, tapez la commande :

```
> python main.py train
```

avec les arguments :

- d DATASET_DIR Path to the dataset to use for training.
- o OUTPUT_DIR Path to the directory where the model and report will be written.
- e EPOCHS Number of epochs the model will be trained for.
- b BATCH_SIZE Number of epochs the model will be trained for.
- (OPTIONAL) -h show this help message.

Une fois l'entraînement terminé, le dossier de sortie contiendra des rapports sur l'entraînement effectué et un fichier **Model.pt** qui contient le réseau qui vient d'être entraîné.

4.4 Evaluation d'un ensemble d'images

maintenant que le modèle est entraîné, il est temps d'évaluer de nouvelles images.

pour cela placez vos nouvelles images dans un nouveau dossier a la racine de l'application.

Ensuite, tapez la commande :

```
> python main.py eval
```

avec les arguments :

```
-i INPUT_DIR                Path to the directory containing images
                             to evaluate.

-o OUTPUT_DIR               Path to the directory where the predicted
                             masks will be written.

-m MODEL                   Pytorch model (.pt file) to use for
                             evaluation.

(OPTIONAL) -s SUB_IMAGE_SIZE Size of the cropped images to pass
                             to the model (Default 256).

(OPTIONAL) -h              show this help message.
```

Les masques créés sont les prédictions d'emplacement des pigments bleus sur l'image associée.