Projet ingénieur et développement durable - Filière Data Engineering INE1

Classification des déchets en utilisant le Deep Learning

Réalisé par :
AGZOUL Imane
BOUAGOUN Khadija
OUKESSOU Houda

Encadré par :
Mme. TOUNSI Karima

Classification des déchets en utilisant le Deep Learning

Remerciements

Nous tenons à remercier toutes les personnes qui ont pu contribuer de prés ou de loin à ce projet, et tout particulièrement notre encadrante **Mme. TOUNSI Karima**, pour l'aide compétente qu'elle nous'a apportée, pour sa patience et son encouragement.

Son œil critique nous'a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Dédicaces

Nous dédions ce travail à nos très chers parents qui n'ont cessé de nous encourager et de nous soutenir jusqu'au bout, à nos frères et sœurs qui ont cru en nous et à tous nos amis qui étaient toujours à l'écoute.

Table des matières

Table des figures

Glossaire

B

Back-end : il se focalise sur le fonctionnement d'un site Web ou d'une application Web. En bref, il représente ce qui se passe dans les coulisses du développement d'une application.

C

CNN : acronyme de "Convolutional Neural Network" qui veut dire en français réseau de neurones convolutifs .

D

Dataset (jeu de données) : est un ensemble de valeurs où chaque valeur est associée à une variable et à une observation.

Deep learning (apprentissage profond) : est un type d'intelligence artificielle dérivé du machine learning où la machine est capable d'apprendre par elle-même. Il s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de couches de neurones, chacune recevant et interprétant les informations de la couche précédente.

F

Front-end : consiste à convertir des données en une interface graphique , à l'aide de HTML, CSS et JavaScript, afin que les utilisateurs puissent afficher et interagir avec ces données.

I

Intelligence artificielle : est un ensemble de théories et des

techniques développant des programmes informatiques complexes capables de simuler certains traits de l'intelligence humaine tels que le raisonnement et l'apprentissage.

M

Machine learning (apprentissage automatique) :est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données .

R

Réseau de neurones artificiels : est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

Réseau de neurones convolutifs : est un type de réseau de neurones artificiels utilisé dans la reconnaissance et le traitement des images, et spécialement conçu pour l'analyse des pixels.

P

Prétraitement des données : est une technique d'exploration de données qui consiste à transformer des données brutes dans un format compréhensible.

Liste des abréviations

- **CNN** - Convolutional Neural Network
- **HTML** - HyperText Markup Language
- **CSS** - Cascading Style Sheets
- **W3C** - World Wide Web Consortium

Introduction générale et Problématique

Face à des ordures massives, le tri manuel s'avère être une tâche laborieuse et qui, en contrepartie, n'aboutit qu'à la classification d'une quantité très limitée de déchets recyclables et nocifs. Or, l'accumulation de ces derniers est une source majeure de la pollution environnementale, qui plus est, se révèle dangereuse pour la santé de l'être humain si elle n'est pas proprement gérée.

Il est donc important d'avoir un système intelligent de gestion des déchets pour faciliter, ainsi qu'améliorer considérablement le processus de tri des ordures.

A cette fin, nous avons eu l'idée de concevoir un site web à travers lequel il sera possible d'importer n'importe quelle image de déchets puis d'obtenir en sortie la classe correspondante telle que le verre, le plastique, le métal, le papier, etc. Pour le mettre en place, nous nous sommes basées sur un modèle de réseau de neurones convolutifs qui est un outil d'apprentissage automatique et qui nous a servi à réaliser la classification.

Dans le présent rapport, nous détaillerons les différentes étapes qui nous ont permis de réaliser ce site web.

Chapitre 1

Elaboration du modèle

1.1 Technologies utilisées

1.1.1 Environnement de travail : Jupyter Notebook

FIGURE 1.1 – Logo de Jupiter

Un Notebook Jupyter est une application web qui vous permet de créer et de partager des documents contenant du code, du texte descriptif, de la visualisation et des équations. C'est donc un moyen pratique d'explorer les données, de documenter et de partager son travail. Il est constitué d'une liste ordonnée de cellules d'entrées et de sorties et organisés en fonction des versions successives du document. Les cellules peuvent contenir du code, du texte au format Markdown, des formules mathématiques ou des images. Notre choix s'est porté sur cet outil car il est pratique en matière de visualisation des données et de création et mise en œuvre de modèles d'apprentissage automatique.

1.1.2 Langage utilisé : Python

FIGURE 1.2 – Logo de Python

Python est le langage de programmation informatique le plus populaire et le plus utilisé, notamment dans le domaine de la Data Science et du Machine Learning. Sa notoriété est dû entre autres au grand choix de librairies qu'il offre et qui permet aux développeurs d'effectuer des tâches complexes sans pour autant se noyer dans de nombreuses lignes de code. C'est justement pour cette raison que nous avons choisi de l'utiliser dans ce projet. D'ailleurs parmi les librairies que nous avons utilisées, on trouve Pandas et PyTorch qui sont très privilégiés et amplement exploitées dans le monde du Deep Learning.

1.2 Mise en oeuvre du modèle

1.2.1 Importation et pré-traitement des données

Jeu de données

Nous travaillons sur une base de données contenant 2527 images collectées manuellement par Gary Thung et Mindy Yang. Nous cherchons à déterminer la classe d'une image parmi les six catégories suivantes : carton, verre, métal, papier, plastique ou divers.

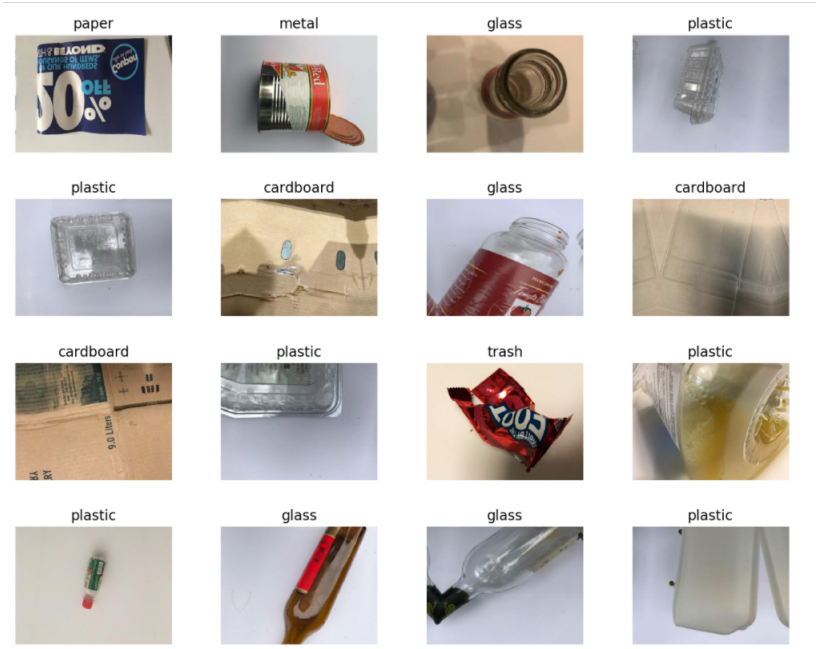


FIGURE 1.3 – Exemples de nos données

1.2.2 Importation des librairies et des données

Importation des librairies

In [1]:
```
import os  #Ce module est une bibliothèque dédié aux besoins de gestion de fichiers et de dossiers.
import torch
import torch.nn.functional as F
from torchvision import datasets, transforms
from tqdm.notebook import tqdm, trange
import numpy as np
import os.path
```

Importation des données

In [4]:
```
#Affichage du dossier de travail
data_dir = 'C:\\Users\\Houda\\Desktop\\GarbageDataSet'
#Affichage des sous-dossiers du dossier de travail
classes = os.listdir(data_dir)
print(classes)
```

```
['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']
```

Importation des images contenues dans data_dir dans la variable dataset tout en les convertissant en "tensor" qui représente le format interne des vecteurs et matrices que Pytorch utilise.

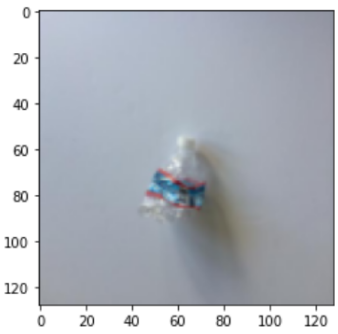
In [4]:
```
dataset = datasets.ImageFolder(data_dir,transform = transforms.Compose([
    transforms.Resize((128,128)),transforms.ToTensor()]))

#Affichage de la taille du dataset:
print(len(dataset))
```

```
2527
```

1.2.3 Exploration et visualisation des images du Dataset

In [7]:
```
import matplotlib.pyplot as plt
%matplotlib inline

def show_sample(img, label):
    print("Label:", dataset.classes[label])
    plt.imshow(img.permute(1, 2, 0)) #Réarrangement du tenseur en prenant en compte l'ordre désiré.

img, label = dataset[2000]
show_sample(img, label)
```

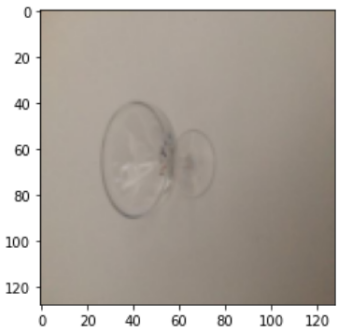
```
Label: plastic
```



In [5]:

```
import matplotlib.pyplot as plt
%matplotlib inline

def show_sample(img, label):
    print("Label:", dataset.classes[label])
    plt.imshow(img.permute(1, 2, 0)) #Réarrangement du tenseur en prenant en compte l'ordre désiré.

img, label = dataset[459]
show_sample(img, label)
```

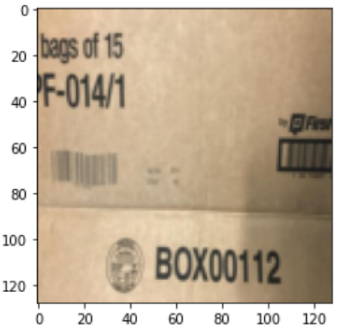
```
Label: glass
```



In [7]:

```
import matplotlib.pyplot as plt
%matplotlib inline

def show_sample(img, label):
    print("Label:", dataset.classes[label])
    plt.imshow(img.permute(1, 2, 0)) #Réarrangement du tenseur en prenant en compte l'ordre désiré.

img, label = dataset[189]
show_sample(img, label)
```

```
Label: cardboard
```



1.2.4 Préparation des jeux de données d'entrainement, de test et de validation et entrainement du modèle

Préparation des jeux de données d'entrainement, de test et de validation

Le dataset contient 2527 images. Nous allons diviser notre dataset en 3 ensembles :

train set (60%) – validation set (20%) – test set (20%)

In [8]:
```
from torch.utils.data import random_split
random_seed = 42
torch.manual_seed(random_seed)
train_set, validation_set, test_set = random_split(dataset, [1517, 505, 505])
```

In [9]:
```
print("Nombre d'images contenues dans le train set: {}".format(len(train_set)))
print("Nombres d'images contenues dans le test set: {}".format(len(test_set)))
print("Nombres d'images contenues dans le validation set: {}".format(len(validation_set)))
```

```
Nombre d'images contenues dans le train set: 1517
Nombres d'images contenues dans le test set: 505
Nombres d'images contenues dans le validation set: 505
```

Ensuite, On utilisera un DataLoader qui se chargera de mélanger et de grouper en lots les données.

In [11]:
```
train_loader = torch.utils.data.DataLoader(train_set, batch_size=1, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=1, shuffle=False)
```

Entrainement du modèle

```
import torchvision.models as models
model = models.resnet18(pretrained = True)

import torch.nn as nn

# Fonction de perte (Loss function) et l'Optimizer :
criterion = nn.CrossEntropyLoss()  # Calculer l'erreur entre la prévision du réseau et la valeur ré
optimizer = torch.optim.Adam(model.parameters(), lr=5.5e-5) #algorithme d'optimisation

for epoch in trange(7):
    for images, labels in tqdm(train_loader):
        # Réinitialisation du gradient :
        optimizer.zero_grad()

        # Forward pass
        x = images
        y = model(x)  #calcul de la sortie du réseau
        loss = criterion(y, labels) #calcul de la perte

        # Backward pass
        loss.backward() # rétro-propagation de la correction d'erreur dans les couches antérieures
        optimizer.step() #correction des poids synaptiques à partir des gradients calculés
```

1.2.5 Mesure des performances du modèle

```
# Test
correct = 0
total = len(test_set)

with torch.no_grad():
    # Itération sur les minibatchs du test set
    for images, labels in tqdm(test_loader):
        # Forward pass
        x = images
        y = model(x)

        predictions = torch.argmax(y, dim=1)
        correct += torch.sum((predictions == labels).float())

print('Test accuracy: {}'.format(correct/total))
```

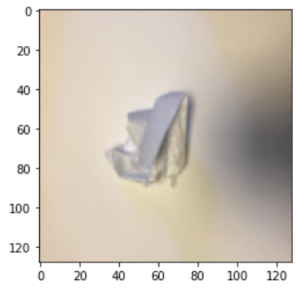
```
Test accuracy:  0.77865425663
```

1.2.6 Visualisation de quelques prédictions d'images prises au hasard dans le test set

Visualisation des prédictions

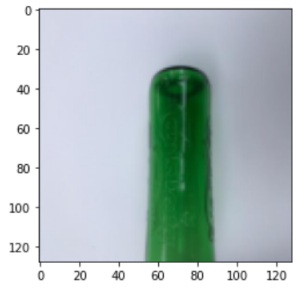
In [26]:
```
def predict_image(img, model):
    # Convertir à un batch de taille 1
    xb = img.unsqueeze(0)
    # Dégager les prédictions à partir du modèle
    yb = model(xb)
    # Choisir l'indice ayant la probabilité la plus élevée
    prob, preds  = torch.max(yb, dim=1)
    # Récupérer l'étiquette de la classe
    return dataset.classes[preds[0].item()]

img, label = test_set[340]
plt.imshow(img.permute(1, 2, 0))
print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))
```

```
Label: paper , Predicted: paper
```




```
img, label = test_set[210]
plt.imshow(img.permute(1, 2, 0))
print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))
```

```
Label: glass , Predicted: glass
```



1.3 Conclusion

Nous avons pu analyser dans ce chapitre en détail l'architecture de notre modèle de la collection des données jusqu'à l'entrai-

nement et l'évaluation de sa performance.

Dans le prochain chapitre on parlera de la mise en oeuvre de l'application web, évoquant les technologies utilisées et une présentation des étapes du déploiment.

Chapitre 2

Déploiement du modèle

2.1 Déploiement du modèle

2.1.1 HTML

FIGURE 2.1 – Logo de HTML

L'HyperText Markup Language, HTML, désigne un type de langage informatique descriptif. Il s'agit plus précisément d'un format de données utilisé dans l'univers d'Internet pour la mise en forme des pages Web. Il permet, entre autres, d'écrire de l'hypertexte, mais aussi d'introduire des ressources multimédias dans un contenu.

Développé par le W3C (World Wide Web Consortium) et le WHATWG (Web Hypertext Application Technology Working Group), le format ou langage HTML est apparu dans les années 1990. Il

a progressivement subi des modifications et propose depuis 2014 une version HTML5 plus aboutie.

L'HTML est ce qui permet à un créateur de sites Web de gérer la manière dont le contenu de ses pages Web va s'afficher sur un écran, via le navigateur. Il repose sur un système de balises permettant de titrer, sous-titrer, mettre en gras, etc., du texte et d'introduire des éléments interactifs comme des images, des liens, des vidéos... L'HTML est plus facilement compris des robots de crawl des moteurs de recherche que le language JavaScript, aussi utilisé pour rendre les pages plus interactives. [1]

2.1.2 CSS

FIGURE 2.2 – Logo de CSS

Les feuilles de style en cascade1, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML.

Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.[2]

2.1.3 Bootstrap

FIGURE 2.3 – Logo de Bootstrap

Bootstrap est un framework CSS et est une compilation de plusieurs éléments et fonctions web-design personnalisables, le tout emballé dans un seul et même outil.

Les développeurs qui utilisent Bootstrap pour la création de leur site web choisissent les éléments qu'ils veulent utiliser avec la certitude qu'ils ne seront pas incompatibles entre eux. En fait, c'est comme un puzzle. Sauf que dans ce puzzle, chaque pièce s'imbrique parfaitement dans les autres, quelle qu'elle soit. Les éléments personnalisables compilés dans Bootstrap sont une combinaison de HTML, CSS et JavaScript.

Et grâce à la magie de l'open-source, Bootstrap s'améliore en permanence : de nouvelles fonctions absolument géniales ont été ajoutées comme l'aspect responsive (adaptation aux tailles d'écrans) ou la très large sélection de plugins jQuery (présents jusqu'à la version 4 de Bootstrap qu'on emploie dans notre projet).[3]

2.1.4 Flask

FIGURE 2.4 – Logo de Flask

Flask est un micro framework web écrit en Python.

Un Web Application Framework ou simplement un Web Framework représente une collection de bibliothèques et de modules qui permettent aux développeurs d'applications Web d'écrire des applications sans se soucier des détails de bas niveau tels que le protocole, la gestion des threads, etc.

Un framework, dites vous simplement que Flask est un ensemble de modules qui vont vous faciliter la programmation de sites web dynamiques .

2.1.5 Heroku

FIGURE 2.5 – Logo de Heroku

Heroku est un PaaS (Platform as a Service) destinée au développement dans le Cloud. L'ensemble de ses services permet de faciliter le développement d'applications plus fiables, de plus

il prend en charge les langages Ruby on Rails, Java, JavaScript, Node.js, Python, Scala et Clojure.

Heroku est connu pour exécuter des applications dans des dynos, des ordinateurs virtuels. Ces derniers peuvent être mis sous tension ou hors tension en fonction de la taille de votre application. On peut les considérer comme des éléments de base malléables pour exécuter votre application. [4]

2.2 Mise en oeuvre de l'application

2.2.1 Front-end

Le tout début de l'application commence par son nom "Garbage Classifier" avec une petite paragraphe introductive :

FIGURE 2.6 – Le tout début de l'application

la partie qui suit contient la case pour télécharger l'image à classifier et le résultat final de l'opération :

FIGURE 2.7 – Le tout bas de l'application

2.2.2 Back-end

Le back-end est la partie servant à gérer tout le contenu du site visible par les visiteurs sur le front-end.

Les principaux éléments pour le développement étaient englober dans un dossier "flaskProject" avec les dossiers/fichiers suivants :

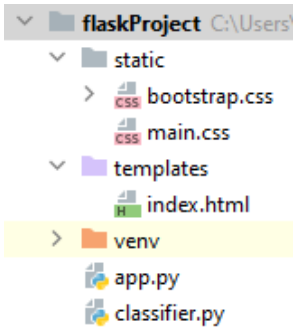


FIGURE 2.8 – Structure des fichiers pour le back-end

- Le fichier **classifier.py** qui contient le code de notre modèle de classification.

- Le fichier **app.py** rassemble :

Les routes qui vont nous permettre de faire le lien entre la requête envoyer par l'utilisateur et la réponse que nous devons renvoyer à l'utilisateur.

le sauvegarde de l'image téléchargée par l'utilisateur.

l'appel à la pages HTML ; en gros Flask utilise Jinja2 comme moteur de templates, cela va nous permettre d'avoir des fichiers lisibles, qui ne contiennent pas du Python.

- Le dossier **templates** qui contient notre template en HTML.

- Le dossier **static** qui contient les deux feuilles de style de notre code HTML et les images utilisées.

- le dossier **env** représentant l'environnement virtuel.

```python
app = Flask(__name__)

UPLOAD_FOLDER = 'upload'

classes = ['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']
transformations = transforms.Compose([transforms.Resize((256, 256)), transforms.ToTensor()])


@app.route('/')
def home() :
    return render_template('index.html')

UPLOAD_FOLDER = 'upload'

@app.route('/', methods = ['GET','POST'])
def upload_predict():
    if request.method =='POST':
        image_file = request.files['image']
        if image_file :
            image_location = os.path.join(UPLOAD_FOLDER, image_file.filename)
            image_file.save(image_location)
            image = Image.open(image_location) #Path('UPLOAD_FOLDER/' + image_file.filename))
            example_image = transformations(image)

            pred = predict_image(example_image, model)
            return render_template('index.html', prediction=pred) # , image_loc = image_file.filename)


if __name__ == '__main__':
    app.run()
```

FIGURE 2.9 – Code de la spécification des routes vers les templates

2.2.3 Déploiment sur Heroku

L'application crée avant ne fonctionne que localement, pour la rendre public on la déploie sur Heroku.

Heroku lit deux fichiers pour configurer l'environnement de production :

Procfile : contient les instructions pour démarrer l'application.

```
web: gunicorn app:app
```

FIGURE 2.10 – Fichier Procfile

requirements.txt : contient les librairies nécessaires pour que le projet fonctionne, pour notre cas ces librairies sont :

```
-f https://download.pytorch.org/whl/torch_stable.html
gunicorn
flask
torch==1.8.1+cpu
torchvision==0.9.1+cpu
```

FIGURE 2.11 – Fichier requirement.txt

Parce que nous avons utlisé notre modèle enregistré avec pickle (a une grande taille), on a eu besoin d'intégrer Git-LFS avec Heroku en ajoutant un buildpack spécifique pour ce propos et une variable de configuration nommée HEROKU_BUILDPACK_GIT_LFS_REPO.

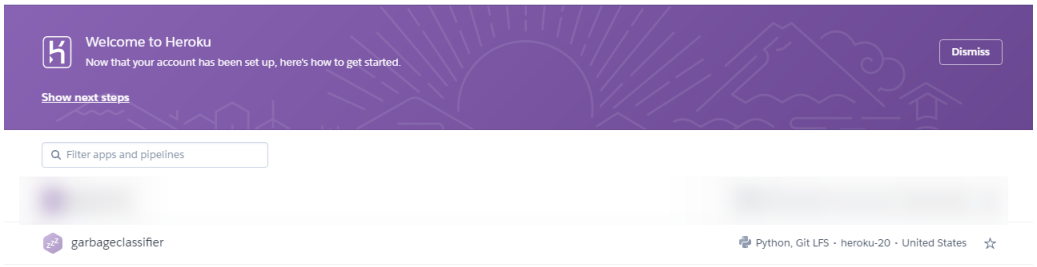


FIGURE 2.12 – L'application web sur Heroku après déploiment

Et voilà le resultat donné après la classification d'un image contenant du plastic :

FIGURE 2.13 – L'image fournie à l'application

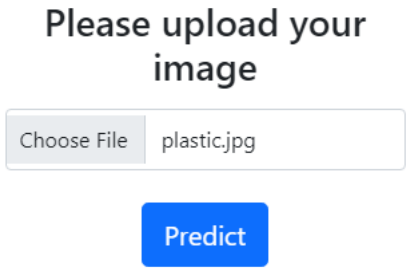


FIGURE 2.14 – Résultat de la classification

2.3 Conclusion

Pour conclure cette partie, le travail que nous avions présenté constitue une base solide et adaptable aux futurs changements ainsi qu'aux perspectives que nous envisageons afin de présenter une expérience utilisateur de la façon la plus optimale.

Perspectives

Pour que notre application web soit accessible et utilisable par tout le monde, on a eu recours à Heroku comme il était minutieusement expliqué dans le deuxième chapitre de ce rapport, et vu que les problèmes techniques sont incontournables pour tout projet, on en a malheureusement aussi été victimes durant le téléchargement de notre modèle pour commencer le déploiement.

En testant notre site web dont le lien est explicité ultérieurement dans le rapport, on s'est rendu compte que les prédictions faites sont un peu aléatoires, contrairement à sa performance dans le localhost de nos machines où notre modèle donne souvent des prédictions raisonnables.

Après maintes recherches sur les forums dédiés au développement web et au Deep Learning, on a constaté que plusieurs développeurs se plaignent du fait que les paramètres de leur modèle se changent par défaut lors de son téléchargement à l'aide de la librairie pickle, et par conséquent le modèle fonctionne bien dans le notebook ou il a été entrainé mais non pas sur un autre notebook où on l'a juste téléchargé, la chose qui explique aussi la différence entre la performance de notre modèle de classification avant et après le déploiement.

Par conséquent, ce problème inatendu qu'on a confronté vers la fin de notre travail rend le site web généré ne fonctionne pas comme on l'a espéré et le facteur du temps le rend encore plus dif-

ficile pour le moment à diagnostiquer le problème pour lui trouver des solutions.

Il est à noter que notre travail ne s'arrête pas ici et qu'on va sûrement résoudre ce petit souci technique pour que notre site web soit fonctionnel comme prévu .

Et afin d'améliorer davantage la performance de notre application de classification de déchets, on va lui ajouter une partie qui serait dédiée à l'utilisateur où il peut réclamer une faute de prédiction, comme il peut aussi enrichir notre dataset en téléchargeant une image d'une classe de déchets tout en spécifiant son étiquette.

Sous toujours la même perspective, on va essayer ultérieurement de modifier les paramètres de votre modèle pour améliorer sa précision et pour qu'il soit plus authentique surtout pour les entreprises spécialisées dans le domaine du recyclage.

Conclusion générale

Le recyclage est conçu comme l'une des astuces les plus bénéfiques à notre environnement. Et pour qu'il soit bien réussi, il nous faut un outil puissant et crédible qui va faire la tâche de classification de déchets à bon escient .

Pour ce faire, on a pensé à automatiser ce processus épuisant par le biais du deep learning, non pas seulement pour faciliter la tâche de classification, mais aussi dans le but de pouvoir classifier les éléments pour lesquels même l'Homme se trouve perplexe et incapable à determiner leur nature .

Quant à la réalisation du projet, il s'avère bien qu'il nécessite une bonne connaissance en termes du Deep learning ainsi que le développement web.

Finalement, ce projet était une opportunité onéreuse pour nous pour s'initier au monde de l'intelligence artificielle, bien notamment le deep learning, comme il nous a fait forger notre sens de responsabilité, le respect des délais ainsi que l'esprit d'équipe qui représentent tous des traits importants pour se faire distinguer dans le monde professionnel.

Références bibliographiques

[1] What is HTML ?
- https ://www.yourhtmlsource.com/starthere/whatishtml.html
[2] Feuilles de style en cascade
https ://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade
[3] What is Bootstrap and How Do I Use It ?
https ://www.taniarascia.com/what-is-bootstrap-and-how-do-i-use-it/
[4] QU'EST-CE QUE HEROKU ? VOICI LE GUIDE COMPLET
https ://www.objetconnecte.com/heroku-tout-savoir/

Annexe

L'application Web : garbageclassifier.herokuapp.com