La reconnaissance des plaques d'immatriculation :

Introduction:

La reconnaissance de plaques d'immatriculation est un domaine de l'intelligence artificielle qui consiste à utiliser des algorithmes pour lire et interpréter les caractères sur les plaques d'immatriculation des véhicules. Ce projet en Python pourrait utiliser des techniques d'apprentissage automatique pour entraîner un modèle capable de reconnaître les caractères sur les plaques d'immatriculation à partir d'une base de données d'images de plaques. Il pourrait également utiliser des techniques de traitement d'image pour détecter la position de la plaque d'immatriculation sur une image avant de lire les caractères.

1. OpenCV-Python :

1.1. DEFINITION :

OpenCV-Python est la bibliothèque Python pour OpenCV, un logiciel de vision par ordinateur et d'apprentissage automatique open-source. Il fournit une interface pratique à la bibliothèque OpenCV en utilisant le langage de programmation Python, facilitant ainsi les développeurs à écrire des applications de vision par ordinateur en Python. OpenCV-Python vous permet d'utiliser une large gamme de techniques de traitement d'image et de vision par ordinateur, telles que la détection d'objets, la reconnaissance d'images et l'analyse vidéo, dans vos applications Python. Il fournit également un grand nombre de modèles pré-entraînés et de codes d'exemple qui peuvent être utilisés pour ajouter rapidement et facilement des fonctionnalités de vision par ordinateur à vos applications Python.

1.2. installer OpenCV-Python :

Pour installer OpenCV-Python sur un ordinateur, vous pouvez utiliser pip, qui est un gestionnaire de paquets Python. La commande suivante permet d'installer OpenCV-Python en utilisant pip :

```
pip install opencv-python
```

Il est également possible d'installer OpenCV-Python en utilisant d'autres gestionnaires de paquets, tels que conda pour les utilisateurs de Anaconda.

```
conda install -c conda-forge opencv
```

Il est important de noter que l'installation peut prendre un certain temps, car OpenCV est une bibliothèque volumineuse qui inclut de nombreux fichiers. Il est également possible de rencontrer des erreurs liées aux dépendances lors de l'installation. Dans ce cas, il peut être nécessaire de les installer manuellement ou de mettre à jour vos versions de python ou les librairies utilisées.

Après l'installation, vous pouvez vérifier si OpenCV est correctement installé en utilisant la commande suivante :

```
python -c "import cv2; print(cv2.__version__)"
```

Cela devrait afficher la version d'OpenCV que vous avez installée.

1.3 OpenCV avantages :

Il y a plusieurs avantages à utiliser OpenCV pour des projets de vision par ordinateur et de traitement d'image :

Fonctionnalités riches : OpenCV fournit un grand nombre de fonctions pour traiter les images et les vidéos, y compris la détection d'objets, la reconnaissance de formes, la segmentation d'image, la correction de distorsion, la stabilisation de vidéo et bien plus encore.

Prise en charge multiplateforme : OpenCV est disponible pour Windows, Linux, MacOS et même les appareils mobiles, ce qui permet de développer des applications pour une variété de plateformes.

Langues de programmation multiples : La bibliothèque OpenCV est écrite en C++, mais il existe également des bindings pour Python, Java, MATLAB et d'autres langages de programmation, ce qui facilite l'intégration dans des projets existants.

Grande communauté : OpenCV est un projet open-source très populaire et est utilisé dans de nombreux projets commerciaux et académiques. Il existe une grande communauté active qui contribue au développement de la bibliothèque et partage des informations et des exemples de codes.

Performances élevées : OpenCV est optimisé pour les performances et peut gérer des images et des vidéos de haute résolution en temps réel.

Modèles pré-entraînés : OpenCV offre un grand nombre de modèles pré-entraînés pour les tâches courantes de traitement d'image et de vision par ordinateur, tels que la reconnaissance faciale et la détection de mouvement, ce qui permet de gagner du temps et des ressources pour les projets.

1.4 OpenCV fonctionnement :

Le fonctionnement d'OpenCV repose sur l'utilisation de ses fonctions de traitement d'image et de vision par ordinateur pour manipuler et analyser des images et des vidéos.

Lecture et écriture d'images : OpenCV permet de lire et d'écrire des images à partir de différents formats de fichiers, tels que JPEG, PNG, BMP, etc. Il fournit également des fonctions pour créer des images vides et pour les stocker sur le disque.

Traitement d'images : OpenCV offre une variété de fonctions pour traiter les images, y compris la modification de la luminosité, le contraste, la netteté, le flou, la rotation, le redimensionnement, etc. Il permet également de détecter des contours, des formes et des couleurs spécifiques dans les images.

Vision par ordinateur : OpenCV offre un grand nombre de fonctions de vision par ordinateur, y compris la détection de visages, de mouvement, de mains, de codes QR, etc. Il peut également être utilisé pour la reconnaissance de formes et la reconnaissance d'objets.

Traitement vidéo : OpenCV permet de lire et d'écrire des vidéos à partir de différents formats de fichiers, et de traiter les images individuelles d'une vidéo pour détecter des objets en mouvement, stabiliser une vidéo ou extraire des images clés.

Apprentissage automatique : OpenCV intègre également des fonctions pour l'apprentissage automatique, comme la régression, les arbres de décision, les réseaux de neurones, etc. Il permet également d'utiliser des modèles pré-entraînés pour la reconnaissance faciale, la détection de mouvement et d'autres tâches courantes de traitement d'image et de vision par ordinateur.

Les fonctions de OpenCV sont généralement appelées en utilisant un enchaînement de commandes pour accomplir une tâche spécifique, cela permet de créer des scripts pour automatiser des tâches répétitives ou pour créer des applications de traitement d'image et de vision par ordinateur en utilisant OpenCV.

1.5 les fonctions opencv utilisée dans notre projet :

Pour traiter les images des voitures on utilise les fonctions suivent :

cv2.imread() : permet de lire une image depuis un fichier **cv2.resize() :** permet de redimensionner une image **cv2.cvtColor() :** permet de convertir une image en niveaux de gris ou de changer son espacement de couleur **cv2.GaussianBlur() :** utilisée pour appliquer un filtre de flou gaussien sur une image. Le filtre de flou gaussien est un type de filtre de flou qui est utilisé pour atténuer le bruit ou les détails indésirables dans une image tout en conservant les contours et les détails importants.

La fonction prend en entrée 3 arguments :

- L'image source (une image en niveaux de gris ou en couleur)
- La taille du noyau (un tuple (x, y) qui spécifie la taille de la matrice de filtrage)
- La deviation standard (un nombre qui spécifie la force du flou)

Cv2.Canny() : utilisée pour la détection de bords dans les images. Il utilise l'algorithme de détection de bords Canny, qui est un algorithme à plusieurs étapes qui détecte les bords en recherchant les zones de l'image avec des valeurs de gradient élevées (c'est-à-dire les zones où l'intensité des pixels change rapidement)

La fonction prend en entrée une image et deux valeurs seuils, et renvoie une image avec les bords mis en évidence en blanc et les non-bords en noir. Les valeurs seuils sont utilisées pour contrôler la sensibilité de la détection de bords et pour éliminer le bruit de l'image.

Cv2.findContours : utilisée pour trouver et extraire les contours des objets dans une image.

Les contours sont définis comme les limites d'un objet dans une image et peuvent être utilisés pour diverses tâches de traitement d'image, telles que la reconnaissance d'objets, la segmentation d'image et l'analyse de forme.

La fonction prend en entrée une image et un mode, et renvoie une liste de contours, chacun représenté par une liste de points.

Le paramètre de mode contrôle le type de contour qui est renvoyé, et peut être cv2.RETR_EXTERNAL, cv2.RETR_LIST, cv2.RETR_CCOMP ou cv2.RETR_TREE. La fonction est généralement utilisée pour la reconnaissance d'objets et l'analyse de forme.

Cv2.arcLength() : Utilisée pour calculer la longueur d'une courbe de contour.

La fonction prend en entrée un contour représenté par une liste de points, ainsi qu'un booléen qui indique si le contour est fermé ou non.

Elle renvoie un nombre flottant qui correspond à la longueur de la courbe de contour.

Cv2.approxPolyDP() : utilisée pour approximer un contour par une courbe polygonale.

La fonction prend en entrée un contour représenté par une liste de points et un paramètre de précision epsilon qui contrôle la précision de l'approximation.

Elle renvoie une liste de points qui représente une courbe polygonale approximative du contour d'entrée.

Cette fonction est utile pour simplifier les contours complexes ou pour identifier les formes géométriques de base dans les images.

Cv2.boundingRect() : utilisée pour obtenir un rectangle englobant qui contient entièrement un contour.

La fonction prend en entrée un contour représenté par une liste de points et renvoie les coordonnées (x, y) de l'angle supérieur gauche ainsi que la largeur et la hauteur du rectangle englobant.

Cette fonction est utile pour localiser les objets dans les images ou pour mesurer les caractéristiques physiques des objets.

Cv2.putText() : utilisée pour ajouter du texte à une image.

La fonction prend en entrée une image, une chaine de caractères qui représente le texte à ajouter, les coordonnées (x, y) où le texte doit être placé dans l'image, le type de police à utiliser, la taille de la police, les couleurs (en RGB ou en gris) et un paramètre de thickness qui définit l'épaisseur des lettres.

La fonction modifie l'image d'entrée en y ajoutant le texte.

Cv2.drawContours() : utilisée pour dessiner les contours d'objets sur une image.

La fonction prend en entrée une image, une liste de contours à dessiner, un indice de contour (pour dessiner un seul contour d'une liste de contours), une couleur et une épaisseur de ligne. Elle modifie l'image d'entrée en y dessinant les contours.

Cette fonction peut être utilisée pour visualiser les résultats de la détection de contours, pour surligner les objets dans une image ou pour créer des images annotées.

Cv2.imshow() : utilisée pour afficher une image.

La fonction prend en entrée le nom de la fenêtre dans laquelle l'image doit être affichée et l'image elle-même. Elle ouvre une fenêtre qui affiche l'image. Cette fonction est utilisée pour visualiser les résultats de traitements d'image ou pour afficher des images pour la vérification manuelle.

Il est important de noter que pour que l'affichage de l'image soit actualisé il faut utiliser **cv2.waitKey()** qui permet de capturer les événements clavier.

2. EasyOCR package :

2.1 DEFINITION :

EasyOCR est une bibliothèque Python pour la reconnaissance optique de caractères (OCR) qui facilite l'utilisation de Tesseract OCR. Il utilise les fonctions de OpenCV pour améliorer la qualité de l'image avant de la soumettre à Tesseract pour une meilleure reconnaissance. Il permet de reconnaître le texte à partir d'images et de vidéos en utilisant des techniques de traitement d'image pour améliorer la qualité de l'image avant de l'envoyer à Tesseract pour la reconnaissance de caractères.

EasyOCR est facile à utiliser, il suffit de charger une image ou une vidéo, de spécifier la zone de texte à reconnaître, et de lancer le processus de reconnaissance. Le résultat sera le texte reconnu. Il est également possible de l'utiliser en ligne de commande ou en utilisant des API pour l'intégrer dans un projet existant.

Il prend en charge plusieurs langues, il est capable de reconnaître le texte à partir d'images avec des polices différentes, des tailles et des styles. Il est également capable de reconnaître le texte à partir d'images avec des distorsions, des bruit et des ombres. Il est également possible de créer des modèles personnalisés pour des langues ou des polices spécifiques en utilisant des données d'entraînement.

2.2 installer EasyOCR :

Pour installer EasyOCR, vous aurez besoin de Python et de quelques bibliothèques supplémentaires. Voici les étapes pour installer EasyOCR sur votre ordinateur :

Installez Python en suivant les instructions de l'installateur pour votre système d'exploitation.

Installez les dépendances en utilisant pip :

```
pip install easyocr
```

Vous pouvez maintenant utiliser EasyOCR dans vos projets en important la bibliothèque :

```
import easyocr
```

2.3 EasyOCR avantages :

EasyOCR a plusieurs avantages en tant que bibliothèque de reconnaissance optique de caractères (OCR) :

Facilité d'utilisation : EasyOCR est facile à utiliser et permet de reconnaître le texte à partir d'images et de vidéos en utilisant des fonctions simples et intuitives. Il permet également d'utiliser des API pour l'intégrer dans un projet existant.

Support de plusieurs langues : EasyOCR prend en charge plusieurs langues, ce qui permet de reconnaître le texte dans ces langues. Il est également possible de créer des modèles personnalisés pour des langues ou des polices spécifiques en utilisant des données d'entraînement.

Traitement d'image : EasyOCR utilise des fonctions de OpenCV pour améliorer la qualité de l'image avant de la soumettre à Tesseract pour une meilleure reconnaissance. Il utilise des techniques de traitement d'image pour améliorer la netteté de l'image, corriger les distorsions, supprimer le bruit et normaliser la luminosité.

Précision de reconnaissance : EasyOCR est capable de reconnaître le texte à partir d'images avec des polices différentes, des tailles et des styles, même dans des images avec des distorsions, des bruit et des ombres.

Open-source : EasyOCR est un projet open-source ce qui permet de s'adapter à des besoins spécifiques.

Flexibilité : EasyOCR est compatible avec les systèmes d'exploitation Windows, Linux, et MacOS, il peut être utilisé avec diverses applications comme les systèmes de gestion de documents, les applications de numérisation, etc.

2.4 EasyOCR fonctionnement :

EasyOCR fonctionne en utilisant une combinaison de techniques de traitement d'image et de reconnaissance optique de caractères (OCR) pour reconnaître le texte à partir d'images. Il utilise les fonctions de OpenCV pour améliorer la qualité de l'image avant de la soumettre à Tesseract pour la reconnaissance de caractères. Voici les étapes générales pour comment EasyOCR fonctionne :

Chargement de l'image : EasyOCR prend en entrée une image ou une vidéo, qui peut être chargée à partir d'un fichier ou d'une webcam.

Prétraitement de l'image : EasyOCR utilise des techniques de traitement d'image pour améliorer la qualité de l'image, comme la correction des distorsions, l'amélioration de la netteté, la suppression du bruit et la normalisation de la luminosité.

Zone de reconnaissance : EasyOCR permet de spécifier une zone de reconnaissance, c'est-àdire une zone de l'image où le texte doit être reconnu.

Analyse de l'image : EasyOCR utilise Tesseract pour analyser l'image et extraire le texte. Il utilise des algorithmes de reconnaissance de caractères pour analyser les images et extraire le texte.

Résultat : Le résultat est le texte reconnu, il peut être affiché à l'écran ou enregistré dans un fichier.

Il est important de noter que la qualité de reconnaissance dépendra de la qualité de l'image, il est donc important de s'assurer que l'image est nette et bien éclairée. Il est également possible d'utiliser des techniques de traitement d'image pour améliorer la qualité de l'image avant de la soumettre à EasyOCR pour une meilleure reconnaissance.

2.5 les fonctions EasyOCR utilisée dans notre projet :

reader = Reader(["en"], gpu=False, verbose=False) :

Elle permet de définir les paramètres pour la reconnaissance de caractères (OCR) tels que les langues cibles, l'utilisation d'une carte graphique (GPU) et le niveau de détail des informations de sortie.

- ["en"] définit que la langue cible pour la reconnaissance de caractères est l'anglais.
- gpu=False désactive l'utilisation d'une carte graphique pour améliorer les performances de reconnaissance de caractères.
- verbose=False désactive la sortie détaillée des informations pour un fonctionnement en mode silencieux.

En somme, cette fonction Reader permet de définir les paramètres pour la reconnaissance de caractères pour la langue anglaise, sans utiliser la carte graphique, et de manière silencieuse.

reader.readtext() : Utilisée pour extraire le texte à partir d'une image en utilisant la configuration de paramètres définie dans la fonction **Reader()**.

Elle prend en entrée une image (sous forme de chemin d'accès, d'objet image PIL, etc.) et retourne le texte extrait de l'image sous forme de chaîne de caractères.

Il est important de noter que la fonction **reader.readtext()** ne fonctionnera que si une configuration de paramètres valide a été définie précédemment avec la fonction **Reader()**.

3. Realisation:

Interface :

Pour cela on a utilisé la bibliothèque tkinter qu'on a déjà étudiée dans notre cours programmation python

Le code :

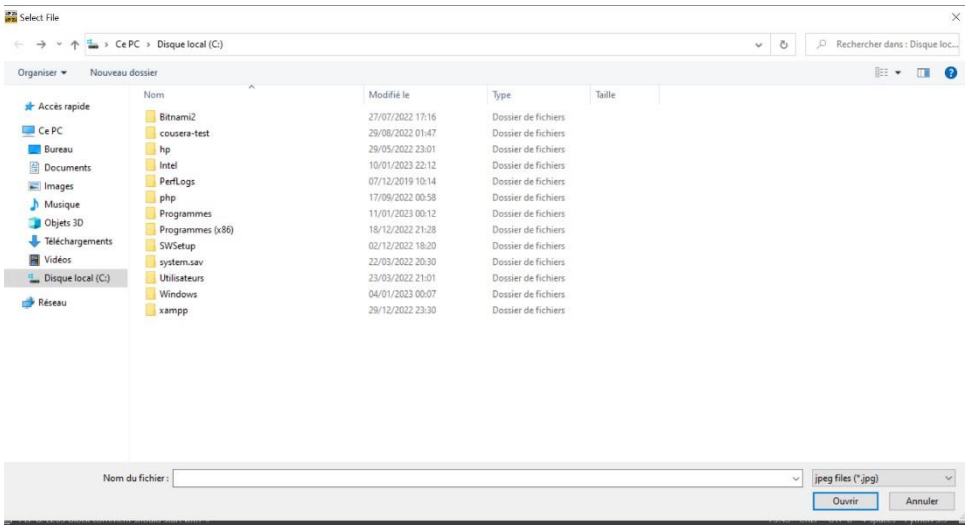
```
root =Tk()
root.title('Matricules')
root.geometry('900x500+300+200')

#LOGO
ig=PhotoImage(file='image/logo.png')
root.iconphoto(False,ig)
root.config(bg='#fff')
root.resizable(False,False)

#IMAGE d'arrière plan

img = PhotoImage(file='image/Mon projet.png')
Label(root, image=img, bg='white').place(x=50, y=50)
#frame
frame = Frame(root, width=350, height=350, bg='white')
frame.place(x=480, y=70)
heading = Label(frame, text='Drag & Drop, \nUpload or Paste image',
fg='#2F86A6', bg='white',font=('Microsoft YaHei UI Light', 15, 'bold'))
heading.place(x=100, y=5)

#button browse
def on_enter(e):
    path_img_txt.delete(0,'end')
def on_leave(e):
    path_img=path_img_txt.get()
    if path_img=='':
        path_img_txt.insert(0,'Entrer a URL')


car_img = Button(frame,
text="Browse",width=15,height=3,bg='#A555EC',command=uploadimg)
car_img.place(x=150,y=70)
```

```
#image path
path_img_txt=Entry(frame,width=40,fg='black',border=2,bg='white',font=('Microsoft YaHei UI Light', 11))

path_img_txt.place(x=30,y=150)
path_img_txt.insert(0,'Entrer a URL')
path_img_txt.bind('<FocusIn>',on_enter)
path_img_txt.bind('<FocusOut>',on_leave)


#submit Button
predict=Button(frame,
width=10,height=2,text='Submit',bg='#472183',fg='white',border=0,command=predict
)
predict.place(x=250,y=200)


#result label
champ_label = Label(frame,border=2,font=('Microsoft YaHei UI Light', 15),
text='')
champ_label.place(x=35,y=270)

#####_____________________________________________________________________________
root.mainloop()
```

1. **Fonction pour importer l'image de machine locale :**

L'évènement de la Button Browse Le code de la fonction :

```
def uploadimg():
    image_name = filedialog.askopenfilename(initialdir="/",
title="Select File", filetypes=(
    ("png files", "*.png"), ("jpeg files", "*.jpg"), ("all files",
"*.*")))
    path_img_txt.delete(0, 'end')
    path_img_txt.insert(0, image_name)
```



1. **Fonction pour la prédiction :**

L'évènement de la Button Submit

Qui fait tout le travail de Charger l'image, Appliquer le filtre de flou gaussien, contourner, lire le texte de matricule image, Le code de la fonction:

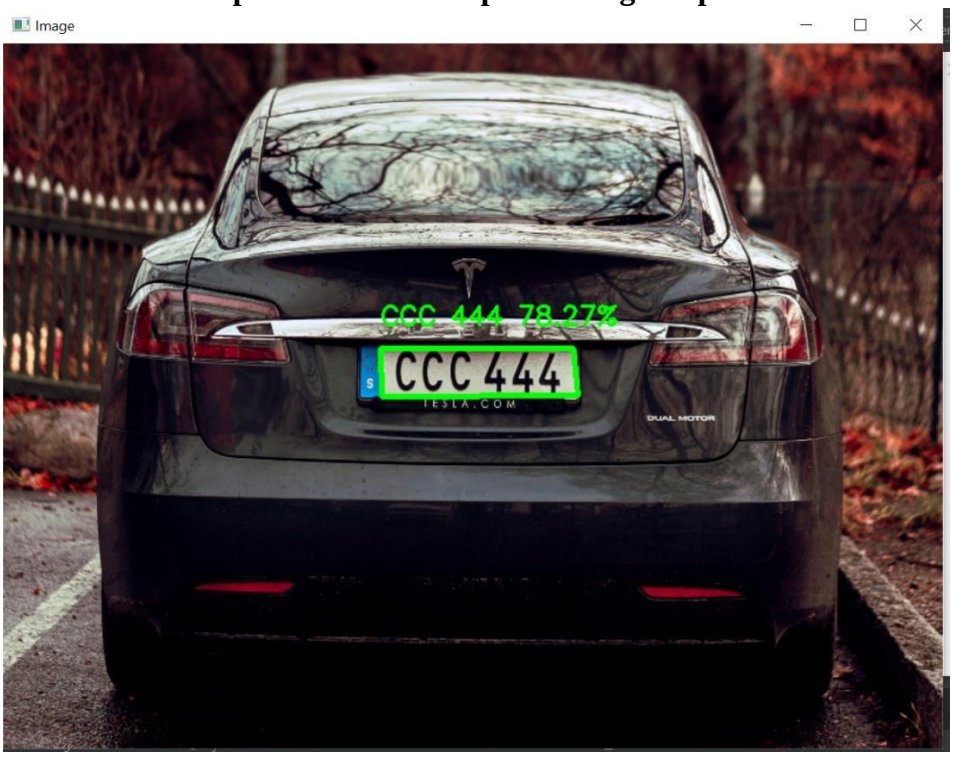
```
def predict():

    print(path_img_txt.get())
    # Charger l'image
    car = cv2.imread(path_img_txt.get())
    #resize the image dimensions
    car = cv2.resize(car, (800, 600))
    gray = cv2.cvtColor(car, cv2.COLOR_BGR2GRAY)
    # Appliquer le filtre de flou gaussien
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    # Edge detection
    edged = cv2.Canny(blur, 10, 200)
    #contours
    cont, _ = cv2.findContours(edged, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cont = sorted(cont, key=cv2.contourArea, reverse=True)[:5]

    for c in cont:
        arc = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.02 * arc, True)
        if len(approx) == 4:
            plate_cnt = approx
            break
    #plate_cnt array
    print(plate_cnt)
    (x, y, w, h) = cv2.boundingRect(plate_cnt)
    #plate
    plate = gray[y:y + h, x:x + w]
    #read text from the plate
    reader = Reader(["en"], gpu=False, verbose=False)
    detection = reader.readtext(plate)
    print(detection)

    if len(detection) == 0:
        text = "Impossible to read the text from \nthe license plate\n****"
        cv2.putText(car, text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255), 3)
        cv2.imshow('Image', car)
        cv2.waitKey(0)
        champ_label.config(text=text,fg='red')
```

```
        else:
        cv2.drawContours(car, [plate_cnt], -1, (0, 255, 0), 3)
        text = f"{detection[0][1]} {detection[0][2] * 100:.2f}%"
        cv2.putText(car, text, (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
        print(text)

       cv2.imshow('license plate', plate)
        cv2.imshow('Image', car)
        cv2.waitKey(0)
        champ_label.config(text=text,fg='#A555EC')

####_____
```

--> image de la matricule :

-->Image de la voiture avec le contour de la matricule et le texte :

--> résultat de la prédiction avec le pourcentage de prédictions :



→Résultat de la prédiction avec le pourcentage de prédiction :


Matricules
Drag & Drop,
Upload or Paste image
Browse
C:/Users/HP/Desktop/image/car1.jpg
Submit
CCC 444 78.27%