Table des matières

[1 DETECTION DE LA PLAQUE D’IMMATRICULATION 2](#_Toc56586891)

[1.1 Ré-entrainer notre propre modèle 2](#_Toc56586892)

[1.1.1 Résultats 4](#_Toc56586893)

[1.2 Utiliser un réseau de neurones pré-entrainé 4](#_Toc56586894)

[1.2.1 Zoom sur le code 4](#_Toc56586895)

[1.2.2 Résultats 5](#_Toc56586896)

# DETECTION DE LA PLAQUE D’IMMATRICULATION

La première étape dans la réalisation de cette application a été d’être capable d’identifier sur une image/photo de véhicule, la plaque d’immatriculation.

Objectif de cette partie :

* Entrée : une photo d’un véhicule pris par l’utilisateur de l’application
* Sortie : la photo de la plaque d’immatriculation du véhicule photographié (ou un message d’erreur s’il n’y a pas de plaque sur l’image)

## Ré-entrainer notre propre modèle

Dans un premier temps, nous avons commencé par travailler avec supervisely et tensorflow, selon les indications du site internet fourni par Damien et Malik.

L’idée principale était d’entrainer son propre réseau de neurones (dont l’architecture en détails était fournie) de manière supervisée grâce au dataset founi par Supervisely.

Grâce à Supervisely, nous avons pu récupérer un dataset contenant 50 images comme celle-ci :

|  |  |
| --- | --- |
| *Image exemple 1* | *Image exemple 2* |
| *Une image contenant bâtiment, extérieur, photo, carrelé  Description générée automatiquement* | *Une image contenant extérieur, bâtiment, photo, bus  Description générée automatiquement* |

Il s’agit finalement de simple montage photo où ils ont collé des images de plaque d’immatriculation sur une image de paysage (buildings, routes, montagnes…).

Chaque image est associée à son « y » :

|  |
| --- |
| *Mêmes images avec un rectangle rouge pour identifier la plaque d’immatriculation* |
|  |

### Zoom sur le code

Ces images étaient donc séparées en train et test sets (44 images / 6 images), puis le modèle était entraîné.

|  |
| --- |
| *Création du modèle* |
| Une image contenant texte  Description générée automatiquement |
| Explications : le modèle est un CNN relativement classique avec des couches de Convolution alternées avec des Max\_Pooling, puis deux couches fully-connected (Dense) et une couche de prédiction (output layer). |

Nous avons réalisé 100 epochs, mais le modèle ne s’améliore plus après une cinquantaine d’epochs et les résultats sont loin d’être aussi bons que ceux présentés sur le site internet.

Une image contenant texte, table

Description générée automatiquement

### Résultats

Les résultats obtenus sont loin d’être satisfaisants :

|  |
| --- |
| *Résultats obtenus sur le test set* |
|  |

Sachant que cette étape était essentielle pour la viabilité du projet de lecture de plaque d’immatriculation, nous avons dans un premier temps, cherché à grossir notre dataset. Malheureusement, nous n’avons rien trouvé, et créer nous-même des images aurait été une solution très longue et pas forcément efficace.

Nous sommes donc partis sur une autre solution :

## Utiliser un réseau de neurones pré-entrainé

Au cours de nos recherches, nous avons trouvé un modèle pré-entrainé, créé par Sergiomsilva[[1]](#footnote-1). Directement, ce modèle est apparu bien plus efficace et complet car :

* Le modèle peut détecter des plaques sur tout type de véhicule (voitures, motos…)
* Le modèle peut lire des plaques du monde entier
* Les résultats sont bien meilleurs – en termes de précision dans la détection et de régularité – que le modèle que nous avons entrainé nous-même (cf. 1.1)

Nous avons donc opté pour cette solution.

### Zoom sur le code

|  |
| --- |
| *Fonction principale (main())* |
|  |
| Explications : Cette fonction principale prend en entrée une image (qui contient supposément un véhicule) et retourne une autre image contenant cette fois-ci uniquement la plaque d’immatriculation. |

|  |
| --- |
| *Fonction color2gray : transforme une image en image nuances de gris* |
| Une image contenant texte  Description générée automatiquement |
| Explications : Cette fonction prend en entrée l’image originale et retourne la même image en nuances de gris (divisé par 255). Cette étape est importante car le modèle utilisé a été entrainé sur des images en nuances de gris.  Nous avons essayé de faire des prédictions sans cette étape, le taux de réussite était très proche de 0%. C’est donc bien une étape essentielle. |

|  |
| --- |
| *Fonction get\_plate : trouve la plaque d’immatriculation sur l’image* |
| *Une image contenant texte  Description générée automatiquement* |
| Explications : Cette fonction prend en entrée l’image du véhicule puis retourne l’image de la plaque d’immatriculation. Pour être parfaitement précis, cette fonction ne trouve pas directement la plaque d’immatriculation, elle fait appel à la fonction detect\_lp() pour cela. |

|  |
| --- |
| *Fonction detect\_lp()* |
| *Une image contenant texte  Description générée automatiquement* |
| Explications : Cette fonction a directement été implémentée par Sergiomsilva pour nous permettre d’utiliser son modèle. Pour cette raison, nous ne l’avons pas modifié et utilisé tel quel.  Nous pouvons voir que c’est bien ici que l’on fait appel au modèle avec model.predict(T). |

### Résultats

Les résultats obtenus sont très satisfaisants.

Voici quelques résultats obtenus avec des photos trouvés sur internet :

|  |  |
| --- | --- |
| *Exemple 1* | |
| *Image originale* | *Image retournée par le programme* |
| *Une image contenant extérieur, route, voiture, rue  Description générée automatiquement* | *Une image contenant arts de la table, assiette, horloge, fermer  Description générée automatiquement* |

|  |  |
| --- | --- |
| *Exemple 1* | |
| *Image originale* | *Image retournée par le programme* |
| Une image contenant extérieur, voiture, bâtiment, route  Description générée automatiquement | Une image contenant extérieur, signe, vert, rue  Description générée automatiquement |

1. <http://sergiomsilva.com/pubs/alpr-unconstrained/?fbclid=IwAR0NTRUedQU3RiXDxzYv8PCvX3OyBsrAdaJSeVaGBZVFpn3CS_4OTPLXdug> [↑](#footnote-ref-1)