Lot STOP

Amaury Colin, Jhon Muñoz

Février 2023

1 Contexte

Le but de l'algorithme est de détecter la présence d'une ligne jaune ou avec des caractéristiques particulières dans l'environnement. Le présent document vise à le formaliser.

2 Données

- I : ensemble des images, on notera $I = I_s \cup I_{ns}$, où $\mathrm{I_s}$ est l'ensemble des images avec une ligne de stop et $\mathrm{I_{ns}}$ l'ensemble des images sans ligne.
- N : nombre d'images (cardinal de I)
- I_i : image i de dimensions (w, h, 3) où w est la largeur, h la hauteur, pour $i \in [\![0, N-1]\!]$. Par défaut, w = 160 et h = 120.
- y_i : entier binaire (0 ou 1) associé I_i pour $i \in [\![0, N-1]\!]$.

$$y_i = \begin{cases} 1 & \text{si } I_i \text{ a une ligne de stop} \\ 0 & \text{sinon.} \end{cases}$$

- f : la fonction indicatrice de I_s telle que $\forall i \in [\![0, N-1]\!], f(I_i) = y_i$.

3 Choix des données :

A priori, notre objectif est seulement de déterminer la présence d'une ligne sans intérêt porté aux observations passées ($\forall i \in [\![0, N-1]\!], f(I_i, I_{i-1}) = f(I_i)$). En ce sens, nous pouvons choisir les images aléatoirement dans notre dataset pour entraîner notre modèle.
Une autre version du modèle pourrait être de déterminer l'apparition de ligne, auquel cas le modèle serait récursif.

À chaque epoch, on choisit de constituer N_b batches de B_s images à l'aide de permutations aléatoires. On note I_j^k l'image $j \in [\![0, B_s]\!]$ du batch k, $k \in [\![0, Nb-1]\!]$.

On considère I équilibré, c'est à dire que $card(I_s) \approx card(I_{ns})$. Si I n'est pas équilibré, on se propose d'augmenter le nombre d'images de l'ensemble I_s ou I_{ns} ayant le cardinal le plus faible par oversampling, c'est-à-dire en lui ajoutant des images transformées du même ensemble (data augmentation). On peut aussi réaliser de l'undersampling en diminuant la taille de l'ensemble de plus grand cardinal.

4 Objectif

On cherche à approximer la fonction f. On note x_i^k la prédiction de $f(Ii)$ par le modèle.

f est à valeur dans $0, 1$. Par conséquent, le choix d'une fonction de perte Binary Cross Entropy semble appropriée. On notera cette fonction de perte l. Ainsi, en notant X^k l'ensemble des prédictions sur les images d'un batch k et Y^k l'ensemble des valeurs attendues, l'objectif du modèle est :

$$\forall k \in [\![0, Nb-1]\!],\ min(l(X^k, Y^k)) = \frac{1}{B_s} \sum_{i=0}^{B_s-1} -\omega_i \cdot (y_i^k \cdot log(x_i^k) + (1-y_i^k) \cdot log(1-y_i^k))$$

À chaque batch, on essaye d'optimiser la fonction de coût et on applique une rétro-propagation pour ajuster les poids du modèle.

On notera ω_i dans l'expression de la fonction qui représente un poids accordé à chaque classe qui est calculé en fonction de la densité de classe afin de rééquilibrer l'ensemble I s'il n'est pas équilibré. Simplement, on peut définir les poids pour toutes les images sans ligne comme étant $card(I_n s)/card(I)$. On peut définir similairement les poids des images avec ligne.

5 Construction du réseau

Le problème actuel s'inscrit dans les problèmes de type computer vision. On choisit donc une architecture ResNet, typique pour ce genre de sujet (car elle possède nombreuses couches de convolution).

On souhaite que les prédictions soient comprises dans l'intervalle [0, 1]. On rajoute donc une couche dense après le average pooling de sorte à n'avoir qu'un seul neurone et on lui renseigne une fonction d'activation de type sigmoïde (Figure 1).

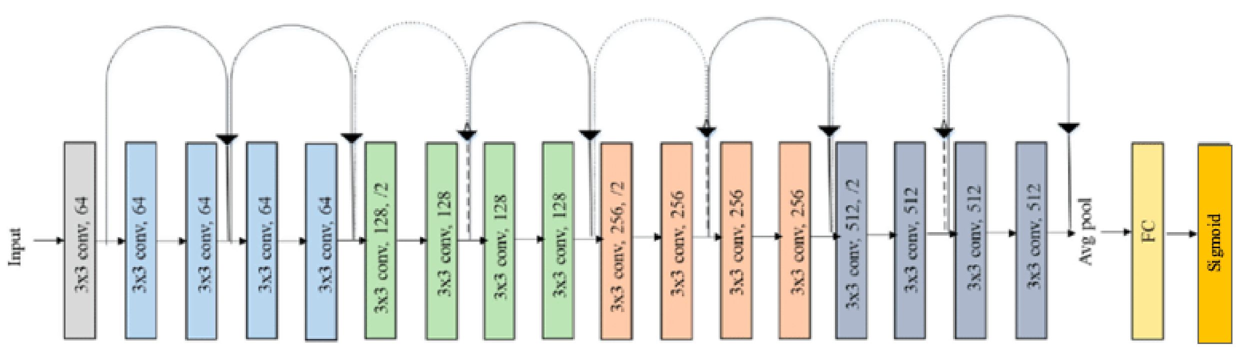

Figure 1: Architecture du réseau

On choisit d'entrainer l'ensemble du réseau à chaque entraînement.

6 Prise de décision

Pour prendre une décision, on se fixe un α correspondant à la marge de certitude que l'on souhaite se fixer pour les prédictions.

Ainsi, pour une prédiction x, la décision d est définie telle que :

$$d = \begin{cases} 1 & \text{si } x > 1 - \alpha \\ 0 & \text{sinon.} \end{cases}$$

Si $d = 1$, on considère qu'il y a une ligne de stop sur l'image, sinon on admet l'absence de ligne.