Outils d'évaluation des incertitudes

Romain Égelé

Août 2017

Table des matières

1 Introduction

Ce projet est mené au sein du laboratoire d'optimisation de l'ENAC et est encadré par Nicolas Durand [1] et Jean-Baptiste Gotteland [2]. Un simulateur du trafic aérien a été réalisé (cf. figure 1). Celui-ci permet de générer des situations entre différents avions, de détecter les conflits propres à cette situation, puis de proposer des trajectoires calculées via un algorithme génétique ou d'effectuer des manoeuvres d'évitement manuellement via une sélection par la souris. Notre but est de donner la main sur le simulateur à de vrais contrôleurs aériens et de pouvoir en extraire des informations, nous nous intéresserons ici aux incertitudes de mesures évaluées "au jugé" par le contrôleur aérien.

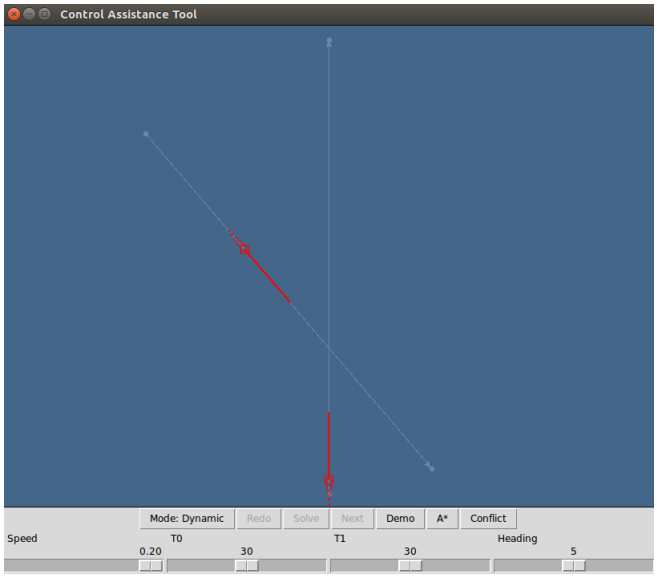

FIGURE 1 – Simulateur

1. enseignant chercheur en optimisation au laboratoire Enac
2. enseignant chercheur en optimisation au laboratoire Enac

1.1 Position du problème

Les incertitudes sont modélisées par 4 paramètres :

— *speed* : l'incertitude sur la vitesse des avions (en pourcentage de la vitesse nominale)
— t_1 : l'incertitude sur le temps de réponse du pilote lors du changement de trajectoire (en secondes)
— t_2 : l'incertitude sur le temps de réponse du pilote lors du retour sur sa trajectoire initiale (en secondes)
— *heading* : l'incertitude sur le cap des avions (en degrés)

Nous aimerions pouvoir deviner les paramètres intuitifs que prennent en compte les contrôleurs aériens lorsqu'ils déclenchent une manoeuvre d'évitement afin de pouvoir comprendre au mieux leurs réactions et leur fournir un logiciel calibré sur leurs attentes. Dans un premier temps, nous allons chercher à deviner le paramètre *speed*.

1.2 Étapes de résolution

Pour résoudre ce problème nous nous positionnerons sur la vision du **Deep Learning** en utilisant des réseaux de neurones à convolution (eng : Convolution Neural Network, CNN). Dans un premier temps nous nous intéresserons à la génération de l'ensemble d'entraînement de notre CNN puis nous l'entrainerons et évaluerons sa performance sur un ensemble d'exemples sur lequel il ne s'est à priori pas entraîné.

1.3 Hypothèses

— Nous considérons ici seulement des problèmes à deux avions (qui sont les plus fréquents).
— Notre logiciel de simulation dispose de 4 paramètres sur les incertitudes, nous décidons de fixer le triplet $(t1, t2, heading) = (30, 30, 5)$, qui sont les valeurs médianes de nos échelles.
— Pour ne pas dupliquer les situations équivalentes par rotation et symétrie nous décidons de fixer la trajectoire de notre premier avion et de faire partir le second avion dans un intervalle d'angle de $[0; \pi]$.

1.4 Choix techniques

Nous avons décidé de résoudre ce problème à l'aide de la library opensource MXNET [1] que nous utiliserons en python 3.5, cette library permet une utilisation rapide des CNNs pour les phases d'apprentissages, de plus elle a l'avantage de tirer parti de la répartition de calcul sur GPU ce qui accélère le temps de calcul pour des algorithmes du type de la stochastic gradient descent où la même opération est répétée sur tous les éléments d'une même matrice.

2 Choix du CNN

On utilise un réseau de neurone éprouvé (cf. figure 2 et figure 4) , [2] sur des images de format $(224, 224)$. Ce réseau de neurones est composé de groupes de couches appelés **inception** et applique la "batch normalization".

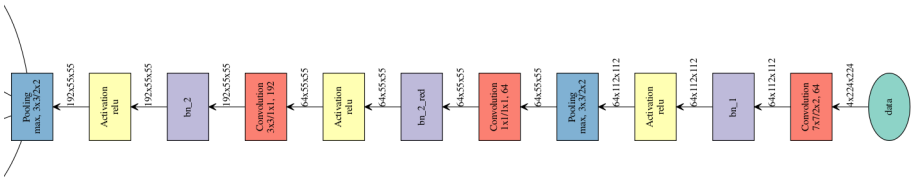

FIGURE 2 – Couches d'entrées du CNN

La sortie (cf. figure 3) du CNN est une FullConnected avec 3 neurones qui ont respectivement pour sortie y_0, y_1, y_2 suivie d'un opérateur Softmax où $softmax(y)_i = \frac{exp(y_i)}{\sum_{j=0}^{n=2} exp(y_j)}$. Ceci nous permet de définir nos classes suivant les incertitudes sur la vitesse qui sont $(0.0, 0.1, 0.2)$ en pourcentage de la vitesse nominale et d'avoir une vision probabiliste du résultat.

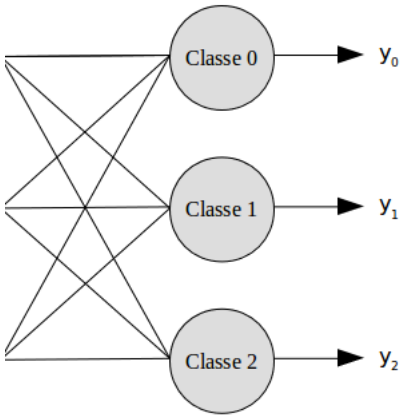

FIGURE 3 – Couches de sortie, FullConnected

Lorsque nous demandons une prédiction au CNN sa réponse correspond à la classe indiquée par $indice\_max(softmax(y))$.

FIGURE 4 – Représentation symbolique du CNN [2]

3 Ensemble d'entraînement

Pour générer l'ensemble d'entraînement de notre CNN nous utilisons le simulateur.

3.1 Première génération

Au cours de la première génération nous parcourons (i_0, i_1, i_2) trois intervalles différents, chaque intervalle est subdivisé en un nombre différent de parts p_0, p_1, p_2. Le premier $i_0 \in [\pi/9; \pi]$ correspond à l'angle de la trajectoire du deuxième avion par rapport au premier, le second $i_1 \in [0; max_t/p_1]$ correspond au temps de départ du premier avion et le troisième $i_2 \in [0; max_t/p_2]$ correspond au temps de départ du second avion. L'algorithme de génération des exemples consiste à parcourir ces intervalles, à chaque situation une évaluation du conflit est effectuée, s'il n'y en a pas on passe à la situation suivante par contre si un conflit est détecté il est résolu via l'algorithme génétique, si la résolution échoue on passe à la situation suivante sinon une image qui fusionne les trajectoires pré-résolution et post-résolution est sauvegardée (cf. figure 6). Cette image suit la nomenclature suivante (cf. figure 5) :

- numéro de l'image générée
- incertitude sur la vitesse
- angle de la trajectoire du second avion
- différence de vitesse entre les deux avions
- décalage temporel sur le départ entre les deux avions
- valeur d'initialisation du hasard

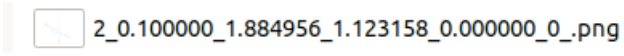

FIGURE 5 – Nomenclature des images

L'image est donc sauvegardée avec sa taille par défaut (suivant la taille du canvas de notre fenêtre) puis nous la redimensionnons afin que ses dimensions soient compatibles avec notre CNN. Les exemples générés dépendent également de paramètres choisis au hasard tels que la vitesse des avions (v_{av1}, v_{av2}) et un bruit que nous appliquons sur l'angle b_{angle}. De ce fait, à chaque génération d'un ensemble nous choisissons une incertitude sur la vitesse $v_{inc} \in \{0.0, 0.1, 0.2\}$ et nous initialisons un hasard de façons différentes pour ne pas obtenir les mêmes exemples.

3.2 Seconde génération

Nous avons ensuite décidé de générer nos exemples suivants plus de paramètres en parcourant toujours différents intervalles i_i divisés en plusieurs parts p_i :

- l'angle du second avion, α.
- le temps de départ du premier avion, t_{dep1}.

— le temps de départ du second avion, t_{dep2}.
— la vitesse du premier avion, v_{av1}.
— la vitesse du second avion, v_{av2}.

Nous avons retiré le bruit aléatoire appliqué sur le cap de l'avion. Chaque situation est évaluée selon les 3 incertitudes sur la vitesse $v_{inc} \in \{0.0, 0.1, 0.2\}$. Nous ne retenons pas les situations conflictuelles non résolues par l'algorithme génétique. Nous gardons les situations lorsqu'elles ont été résolues. Nous gardons également les situations qui sont non conflictuelles avec l'incertitude courante mais conflictuelles avec une incertitude plus élevée (pour montrer l'influence de l'incertitude).

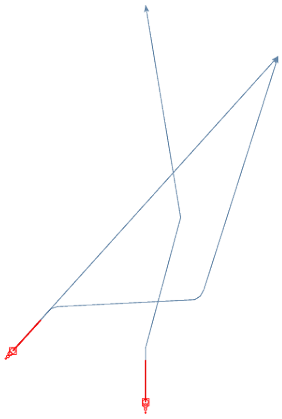

FIGURE 6 – Exemple d'image générée

3.3 Résultats

3.3.1 Première génération

Nous choisissons comme paramètres de génération : $(p_0, p_1, p_2) = (9, 9, 9)$ et $(i_0, i_1, i_2) = (1, 0, 0)$. Nous parcourons chaque incertitude trois fois avec des hasards différents, nous générons donc neuf groupes pour l'ensemble d'entraînement puis nous génèrons trois groupes (un par incertitude) pour l'ensemble de validation. Notre ensemble d'entrainement est alors composé de 3358 situations et notre ensemble de validation de 1077 situations.

Lors de l'apprentissage, le réseau de neurones apprend complètement la base d'entraînement pourtant sa performance sur les prédictions de l'ensemble de validation ne dépasse pas 50%. Si nous demandons une prédiction au réseau de neurones sur une situation inconnue pour lui, il se trompe une fois sur deux, l'apprentissage ne semble donc pas se généraliser. Une autre possibilité est la capacité du réseau de neurone pour s'adapter aux invariances d'un motif. Si celui si reconnait un triangle, même si la rotation et le zoom varie, sa réponse restera la même, pourtant ce genre de variations nous intéresse car elle montre

un allongement du parcours en temps comme en distance, ce qui indique la plupart du temps l'influence d'une incertitude plus forte.

3.3.2 Seconde génération

Nous choisissons comme paramètres de génération :
— l'angle du second avion, $\alpha \in [\frac{\pi}{9}; \frac{5\pi}{9}]$ avec $p_0 = 10$
— le temps de départ du premier avion, $t_{dep1} \in [0; max\_t]$ avec $p_1 = 5$
— le temps de départ du second avion, $t_{dep2} \in [0; max\_t]$ avec $p_2 = 5$
— la vitesse du premier avion, $v_{av1} \in [0.70 \times max\_speed; max\_speed]$ avec $p_3 = 3$
— la vitesse du second avion, $v_{av2} \in [0.70 \times max\_speed; max\_speed]$ avec $p_4 = 3$

Nous parcourons donc $11 * 6 * 6 * 4 * 4 * 3 = 19008$ situations lors de la génération de la base d'entraînement. La base d'entraînement est apprise à 100% par le réseau mais la base de validation n'a pas encore pu être testée.

4 Modules développés

Le code développé a été déposé sur un dépot github : `https://github.com/Deathn0t/tool_evaluation_aircraft_uncertainties`.

4.1 Image2gather

Rassemble les situations générées via le simulateur sous forme d'un dictionnaire python comportant quatres clefs :
— train_data : liste des images d'entraînement.
— train_label : liste des classes correspondant aux images d'entraînement.
— val_data : liste des images de validation.
— val_label : liste des classes correspondant aux images de validation.

4.2 Generation

C'est dans ce module qu'est défini notre CNN.

4.3 Training

Dans ce module nous chargeons un CNN à partir de *generation* ou alors à partir d'un fichier .json et .params si nous voulons lancer l'entraînement à partir d'un réseau déjà entraîné.

4.4 Prediction

Dans ce module nous avons défini différentes fonctions qui permettent de demander une prédiction sur un fichier de données (correspondant aux diction-

naires python décris précédemment), sur un dossier d'images ou sur une image en particulier.

4.5 Trajectory

Dans ce module nous pouvons réaliser différents graphiques d'analyse de nos dossiers d'images afin de mieux comprendre l'influence des incertitudes suivant les situations.

5 Conclusion

Le réseau de neurone utilisé a pu apprendre complètement nos données d'entraînement. Son manque de performance sur nos données de validation pourraît donc être expliqué par la taille trop petite de notre ensemble d'entraînement. Nous aimerions par la suite pouvoir comparer ces résultats avec un réseau de neurones classique (autrement dit composé seulement de couches du type full-connected) qui apprendrait sur les valeurs intrinsèques du problème et non plus des images (la description géométrique de la trajectoire, les temps de réponse...). Nous aimerions également poursuivre en comparant cette classification par utilisation de CNN avec des méthodes de classification plus classique.

Références

[1] T. A. S. F. (ASF). *A Flexible and Efficient Library for Deep Learning.*

[2] S. Ioffe and C. Szegedy. *Inception + BN, suitable for images with around (224, 224).*