

# Résumé de soutenance de stage

Baptiste Bachelier

Biodiv-Wind est une entreprise créée par Henri-Pierre Roche en 2011, basée sur Boujan-sur-Libron. La société fait des études, développe, installe et exploite un système de détection de faune sauvage en milieu industriel pour réduire les impacts de l'installation et de l'exploitation des parcs éoliens sur la faune aviaire. Depuis 2015, ils ont équipé plus de 700 éoliennes en Europe de l'Ouest. L'offre de Biodiv-Wind se compose de plusieurs produits : SafeWind ; Bird Sentinel ; Audiobat ; Xbird Radar.

SafeWind est un système embarqué dont l'objectif est d'assister le personnel opérationnel à réguler la vitesse des pales d'une éolienne dans l'optique d'éviter les collisions avec les oiseaux. Il utilise des algorithmes de traitement d'images pour détecter les objets animés dans des vidéos prises depuis ses propres caméras. Afin d'améliorer les performances de détection du système, l'entreprise a pour projet de s'appuyer sur des modèles d'Intelligence Artificielle (IA) capables de détecter et/ou classifier les objets animés.

Pour cela, de précédentes équipes ont mené différents travaux. Elles ont utilisé des classifieurs d'images, qui utilisent des *Convolutional Neural Network* (CNN). Les modèles d'IA pré-entraînés ont reçu un affinage (*fine-tuning*) pour les spécialiser sur les images de Biodiv-Wind.

A partir des travaux des précédentes équipes, j'ai cherché de nouveaux types de données et de modèles permettant d'améliorer la précision des classifieurs actuels.

Dans un premier temps, je présenterai les activités que j'ai mené au sein de l'entreprise. Dans un second temps, j'aborderai les Réseaux de neurones récurrents (*Recurrent Neural Network*, RNN) qui ont constitué l'essentiel du stage.

Au cours de ce stage, j'ai cherché les informations laissées par les anciennes équipes. La documentation et les prototypes étaient stockés sur les dépôts de l'entreprise ainsi que sur des serveurs spécialisés. J'ai regroupé les informations des anciens projets sur un NAS. Cela a pris beaucoup de temps car il y avait plusieurs téraoctets de données réparties dans plusieurs millions de fichiers et les données étaient dispersées sur les différents serveurs.

Ensuite, en essayant de comprendre les modèles qui ont été testés, j'ai recherché leurs structures et les explications sur le choix des données d'entraînement. Par exemple, pour un des classifieurs basé sur ResNet un *Convolutional Neural Network* (CNN), les données d'entraînement étaient composées d'images mosaïques (Figure 1). L'image est une fusion de l'original avec ses variantes zoomées.



Figure 1 : image mosaïque d'un oiseau

Pour chaque image mosaïque, le modèle apprend les motifs pour des tailles d'objet différents sans nécessité l'augmentation de la taille des données. Ainsi, ce format de données permet d'entraîner le modèle à devenir invariant au changement de taille, sans augmenter le temps d'entraînement.

Avec les travaux qui ont déjà été menés et les informations que j'ai cherchées avec internet, j'ai pu représenter, dans un diagramme (Figure 2), la liaison entre les données et les modèles adaptés à ces données.

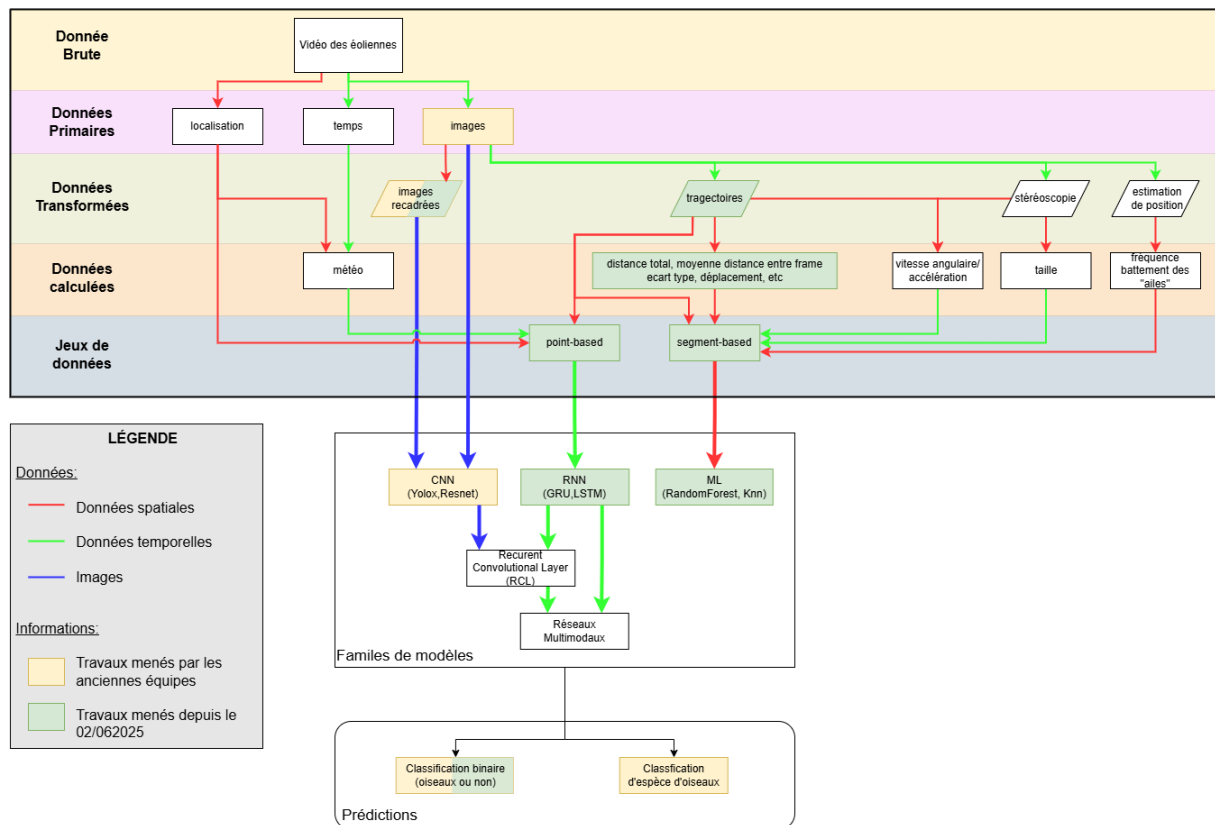


Figure 2 : Diagramme des données et des modèles pour la classification d'objets animés.

Pour entrainer mes prototypes, j'ai créé des jeux de données. Puis j'ai analysé les résultats de leurs entrainements pour pouvoir les comparer et présenter les meilleures architectures pour faire de la classification d'objets animés. Pour les présentations, j'ai appris à faire des schémas « clairs », et à organiser la forme des documents.

Les prototypes avec lesquels j'ai le plus expérimenté sont les *Recurent Neural Network* (RNN).

Les RNN sont un type de modèle équipé d'une mémoire à court terme. Cette mémoire permet de retenir un vecteur caractéristique des élément passés. Par exemple dans la phrase « le chat mange la souris », il va apprendre à retenir un vecteur expliquant le sens : le chat est le prédateur, la souris est la proie.

Mais le RNN de « base » (réseau Elman / Jordan) à un gros défaut, il est sensible aux problèmes de disparition (*vanishing*) et d'explosion (*exploding*) des gradients<sup>1</sup>. Cela a pour conséquence d'empêcher le modèle d'apprendre correctement.

J'ai donc utilisé deux variantes, un *Long Short Term Memory* (LSTM) et un *Gated Recurent Unit* (GRU). Elles résolvent ces problèmes en ajoutant des portes qui filtrent la quantité de données sauvegardée. Je me suis servi de ces variantes pour créer plusieurs

<sup>1</sup> Un gradient est vecteur représentant la variation d'une fonction au voisinage d'un point donné.

modèles entraînés sur des séquences d'une dizaine de positions x et y. Les meilleurs modèles ont pu atteindre les 70% de Précision avec 933 paramètres pour les LSTM et 789 pour les GRU.

Dans un second, temps j'ai trouvé une architecture (Figure 3.a) qui permet d'obtenir les mêmes performances tout en divisant le nombre de paramètres par au moins deux (70% de Précision avec 193 paramètres au lieu de 933 pour le modèle basé sur un LSTM).

La différence se situe sur la forme des données. La trajectoire (série de coordonnées x, y) en entrée est séparée en une série x et une autre y. Chaque série passe dans un RNN (LSTM ou GRU) différent puis on concatène leurs sorties. Ce nouveau vecteur est ensuite passé dans une couche de neurones qui prédit une classe : *Bird* ; *Plane* ; *Insect*.

Une autre version (Figure 3.b) consiste à passer les 2 séries dans le même réseau A. Cela a pour conséquence de diminuer la précision, 70% au lieu de 71%, mais permet de diviser le nombre de paramètres du modèle par deux.

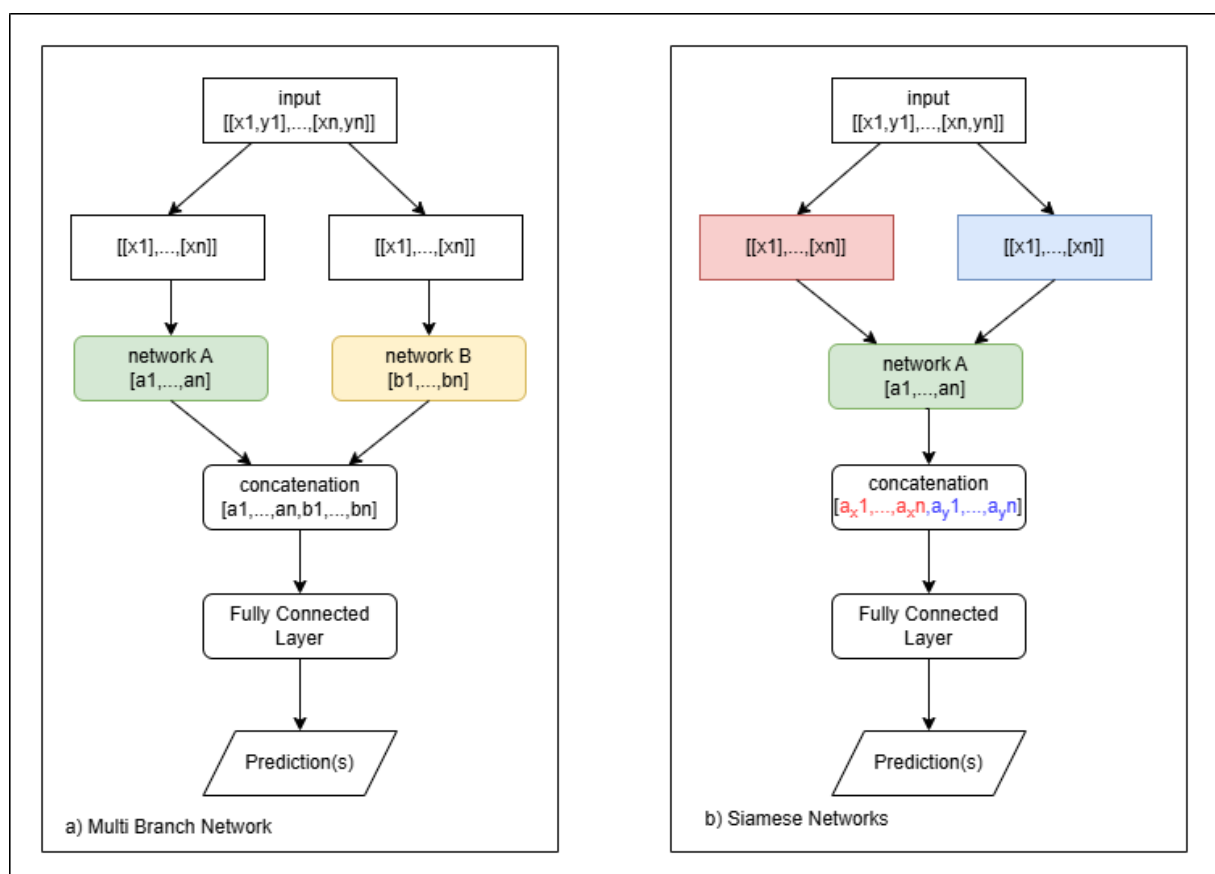


Figure 3 : schéma des architectures Multi Branch (a) et Siamese Networks (b)

Cette différence de précision est causée par l'algorithme d'apprentissage. La descente de gradient permet de trouver l'erreur commise par chaque poids<sup>2</sup> grâce à chaque entrée passée et aux dérivés partielles des fonctions mathématiques du réseau.

Ensuite chaque erreur est soustraite à son poids respectif. Dans notre cas comme deux entrées passent dans le même réseau celui-ci doit être mis à jour deux fois. Parfois, les gradients vont dans des sens contraires, ce qui réduit son apprentissage et cause cette différence de précision.

De plus, dans les réseaux multibranches il semblerait que les variations en x prennent le pas sur celles en y. Cela est à priori dû aux données qui sont majoritairement des trajectoires « horizontales ».

Au cours de ce résumé, j'ai présenté les différentes activités que j'ai mené au sein de l'entreprise comme la création de jeux de données, la programmation de prototypes de modèles d'IA et la rédaction de documentations. Puis J'ai présenté les *Récurrent Neural Network* et leurs utilisations pour de la classification à partir des trajectoires.

Ce stage m'a permis d'acquérir de nouvelles connaissances sur l'intelligence artificielle (IA), tel que le fonctionnement des réseaux résiduels (ResNet) et récurrents (Elman/Jordan).

Je tiens à remercier l'entreprise Biodiv-Wind de m'avoir accueilli au sein de son équipe durant ce stage, en particulier Mr Ergalant pour m'avoir fait confiance et m'avoir intégré dans son équipe. Cette collaboration a été enrichissante et a apporté, je l'espère une valeur ajoutée au projet de la société. Je suis content de pouvoir poursuivre l'aventure en alternance dès septembre.

---

<sup>2</sup> Paramètres appris qui se superposent à l'architecture du modèle pour produire une sortie à partir d'une entrée donnée