

SYS843-01 RÉSEAUX DE NEURONES ET SYSTÈMES FLOUS (H2019)

DÉPARTEMENT DU GÉNIE DE LA PRODUCTION AUTOMATISÉE

---

## **ÉTUDE EXPÉRIMENTALE : RECONNAISSANCE D'ESPÈCES ANIMALES**

---

Soumis par

Hayat ANKOUR

Département du Génie de la Production Automatisée

École de Technologies Supérieures

Montréal, QC

Soumis à

Ismael BEN AYED

École de Technologies Supérieures

Montréal, QC



Le génie pour l'industrie

## Table des matières

<b>1 Mise en situation</b>	<b>3</b>
1.1 Domaine d'application . . . . .	3
1.2 Problématique . . . . .	3
1.3 Objectif du projet . . . . .	4
1.4 Structure du document . . . . .	4
<b>2 Bref sommaire des techniques étudiées</b>	<b>5</b>
2.1 Transfer Learning avec le réseau neuronal convolutif . . . . .	5
2.2 You Only Look Once . . . . .	7
<b>3 Méthodologie expérimentale</b>	<b>9</b>
3.1 Description des bases de données . . . . .	9
3.2 Protocole d'évaluation et mesures de performances (qualités et ressources) . .	9
3.2.1 CNN . . . . .	9
3.2.2 YOLO . . . . .	10
<b>4 Résultats de simulations</b>	<b>11</b>
4.1 Représentation des résultats . . . . .	11
4.1.1 CNN . . . . .	11
4.1.2 YOLO . . . . .	15
4.2 Interprétation des résultats . . . . .	15
4.2.1 CNN . . . . .	15
4.2.2 YOLO . . . . .	16
<b>5 Conclusion</b>	<b>17</b>

# 1 Mise en situation

## 1.1 Domaine d'application

Le but du projet est la reconnaissance d'espèce d'animaux dans une image, puis dans une vidéo si le temps le permet. Il existe plusieurs domaines d'application pour ce type d'algorithme et les informations nécessaires à la reconnaissance des espèces d'animaux comme la taille, la couleur de la fourrure ou encore la position de l'animal dans l'image dépendent du domaine d'application. Par exemple, en forêt, où le nombre de parasite autour de l'animal comme les arbres, buisson et autre est très important, on prendra des caractéristiques bien différentes que sur la banquise, ou l'on ne voit que l'animal sur un fond presque blanc.

L'importance de ce genre d'algorithme dépend également grandement des personnes qui l'utilisent. Il peut être utilisé à des fins de loisirs comme, par exemple, compter le nombre d'espèces croisées lors d'un safari, ou pour des applications telles que la surveillance du nombre d'individus d'une espèce en voie d'extinction. Les applications sont donc diverses et variées.



**FIGURE 1:** Photo de différentes espèces d'animaux dans des milieux environnementaux variés [1]

## 1.2 Problématique

L'identification d'espèce est un défi technologique complexe. De nos jours, ce sont des experts qui identifient et recensent les espèces. Cependant, cela prend beaucoup de temps, et les experts ne peuvent pas effectuer leur travail à grande échelle. De plus, c'est un travail visuel et manuel fastidieux et fatigant.

Une des solutions est d'assister l'expert par l'utilisation d'algorithmes. Donnée une image, comment identifier l'espèce des animaux présents sur l'image?

Une rapide recherche dans la littérature nous permet d'estimer que la méthode la plus utilisée est le réseau neuronal convolutif (Convolutional Neural Network, CNN). Cependant, celui-ci ne permet de faire que de la classification. Un CNN permet d'attribuer une classe à une image. Il faut donc faire un pré-traitement sur les images et obtenir les régions d'intérêts à envoyer au CNN, qui pourra classifier les différentes régions d'intérêt. Dans la littérature, il semblerait qu'une technique largement utilisée soit la recherche sélective, ou l'utilisation d'un autre réseau de neurones qui prédit la position et la taille d'une région d'intérêt.

Mon projet se découpe donc en deux problématiques :

1. Comment déterminer efficacement les régions d'intérêt des différents animaux dans une image?
2. Comment correctement classifier ces images?

Pour mon étude expérimentale, je me suis focalisée sur la seconde problématique.

### 1.3 Objectif du projet

Pour être efficace, il faut entraîner l'algorithme avec un grand nombre d'images. Cela lui permet de reconnaître un maximum d'espèces animales. Pour maximiser l'efficacité de l'algorithme, avoir des images de plusieurs profils différents est nécessaire.

Après l'entraînement de l'algorithme, il faut lui faire un test de validation qui permettra de légitimer les résultats obtenus et, dans le cas contraire, l'entraîner de nouveau afin d'augmenter la précision de classification de l'algorithme.

Il existe de nombreuses références sur la reconnaissance d'espèces animales à l'aide de la méthode CNN comme :

- <https://arxiv.org/abs/1603.06169> [2]
- <https://ieeexplore.ieee.org/abstract/document/7025172/> [3]

### 1.4 Structure du document

J'ai décidé d'étudier le réseau de neurones convolutifs, une des techniques les plus utilisée sous diverses formes (1 à plusieurs couches convolutives) et combinée avec d'autres composantes pour former des algorithmes très connus comme Faster R-CNN ou encore YOLO. J'ai donc pour cela recréer un CNN avec différentes étapes pour mieux comprendre celui-ci sur une base de données comportant 2 classes : chien et chat.

Je vais donc, tout d'abord, dédier la première partie de mon étude expérimentale à celui-ci. En effet, je vais commencer par expliquer le CNN que j'ai utilisée et entraînée dans le domaine du Transfert Learning (que j'expliquerai dans la section 2.1). Je vais vous expliquer les différentes méthodes que j'ai utilisées afin d'arriver au résultat que j'exposerai dans la partie 4.1.

Enfin, je vais vous exposer les résultats que j'ai obtenu à l'aide l'algorithme YOLO sur la base de données Chien et Chat. Cet algorithme met en pratique l'utilisation de CNN et c'est pour cela que j'ai voulu l'étudier avec la même base de données que le CNN étudiée précédemment.

## 2 Bref sommaire des techniques étudiées

Le but final de mon projet est la compréhension des réseaux neuronaux convolutifs que j'ai pu longuement expliquer lors de ma synthèse de littérature. Ce sont des algorithmes très complexes comportant plusieurs couches neuronales imitant le cerveau humain. Chaque couche neuronale permet de reconnaître des caractéristiques qui sont plus ou moins complexes.

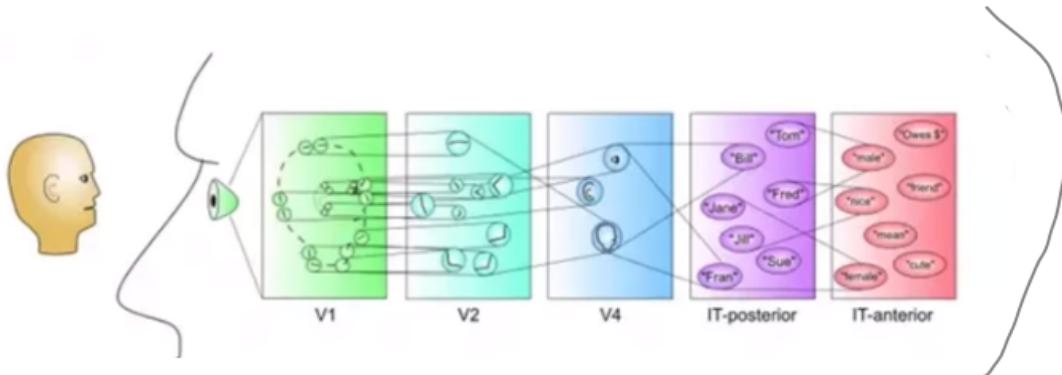


FIGURE 2: Constitution des clusters neuronaux chez l'humain [4]

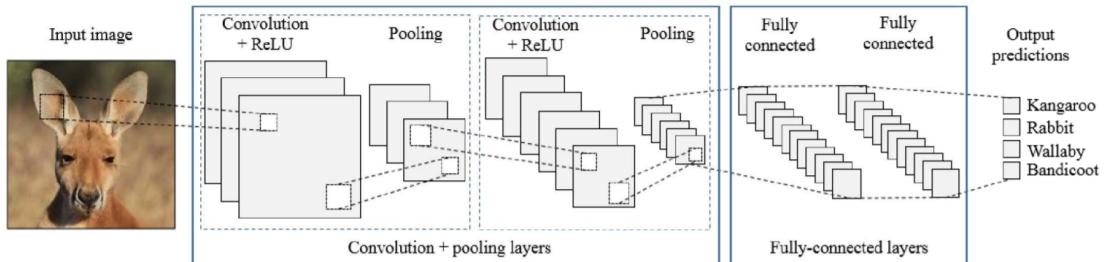


FIGURE 3: Illustration d'une architecture typique d'un réseau neuronal convolutif [5]

- Les techniques que j'ai utilisés lors de mon étude expérimentale sont donc de deux types :
- Le CNN que j'ai étudiée afin de mieux comprendre le fonctionnement des réseaux de neurones convolutifs et la notion de Transfer Learning, une notion que j'ai rencontrée pendant mes recherches et qui m'a semblé importante d'étudier, avec la base de données Chat et Chien à 2 classes.
  - L'algorithme pré-entraîné YOLO, composée d'un CNN, avec la base de données Chat et Chien à 2 classes.

### 2.1 Transfer Learning avec le réseau neuronal convolutif

La meilleure manière de comprendre le fonctionnement d'un réseau neuronal convolutif est de le recréer. J'ai donc suivi un tutoriel[6] permettant d'en recréer un et d'utiliser la technique du Transfert Learning.

**Mais tout d'abord, qu'est-ce que le Transfert Learning?** C'est l'idée de surmonter le paradigme d'apprentissage isolé et d'utiliser les connaissances acquises pour une tâche donnée pour résoudre des tâches du même type. C'est tout simplement le concept des réseaux pré-entraîner. Cette notion est très importante pour les réseaux de neurones car elle permet de gagner beaucoup de temps pour la validation et le test d'un réseau de neurones.

Le modèle de CNN que j'ai utilisé repose sur cette notion. Il est essentiellement composé de 4 couches convolutives :

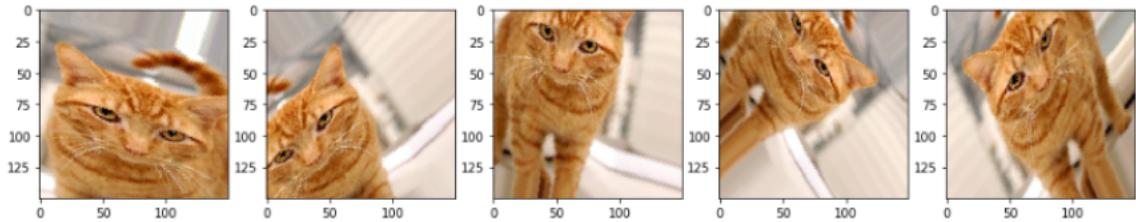
1. La **première couche** de convolution prend une image **500 \* 500 en entrée** et la fait passer à travers 16 filtres. On a en sortie **16 feature maps de taille 498 \* 498**. Il y a ensuite une couche de **MaxPooling** de sous-échantillonage qui donne en **sortie 16 feature maps de taille 249 \* 249**.
2. La **deuxième couche** de convolution retourne **64 feature maps de taille 247 \* 247**. Le sous échantillonne est effectué par une couche de **MaxPooling** qui donne en **sortie 64 feature maps de taille 123 \* 123**.
3. La **troisième couche** de convolution retourne **128 feature maps de taille 121 \* 121**. Encore une fois, on sous échantillonne avec une couche de **MaxPooling** qui donne en **sortie 128 feature maps de taille 60 \* 60**.
4. La **quatrième couche** de convolution retourne **128 feature maps de taille 58 \* 58**. On sous échantillonne avec une couche de **MaxPooling** qui donne en **sortie 128 feature maps de taille 29 \* 29**.

Il comporte aussi quatre autres couches :

- Une couche **flatten** qui est utilisé pour redimensionner le résultat donné en sortie de la couche 4 vers un **vecteur 1 \* 107648**.
- Une couche ReLU qui récupère le résultat de la couche flatten.
- Une **couche de dropout afin de régulariser** le dataset pour éviter un overfitting dû à la mauvaise qualité du dataset.
- Enfin, la dernière couche qui prédit la classe de l'image à l'aide de la **fonction d'activation Softmax**.

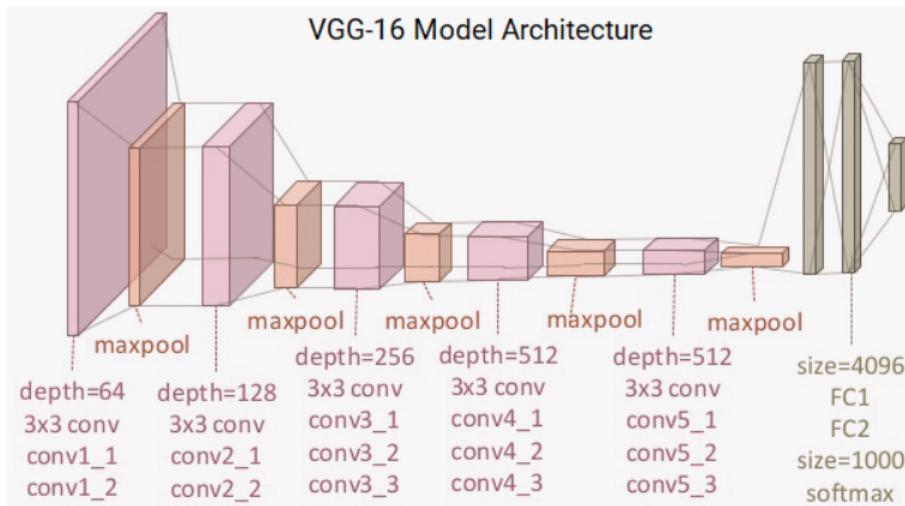
Afin d'étudier aux mieux les effets d'un CNN, j'ai utilisé 6 modèles différents :

1. J'ai d'abord implémenté le CNN sans régularisation, à partir de zéro ("from scratch"), afin de voir son effet direct avec la base de données que je lui ai envoyé.
2. J'ai ensuite utilisé une régularisation après les résultats obtenus. J'exposerai et j'expliquerai donc les résultats dans la partie 4 de ce document.
3. La base de données étant petite (voir 3.2), j'ai fait une augmentation de données à l'aide de la librairie Keras, afin de générer des images aléatoirement à partir des images du dataset et d'avoir beaucoup plus de données à étudier.



**FIGURE 4:** Résultat issu de la génération aléatoire d'image à partir de celle d'un chat

4. Afin d'étudier le Transfer Learning, j'ai utilisé le modèle VGG-16 avec la base de données simple Chat et Chien. En effet, le modèle VGG-16 est un modèle comportant 16 couches convolutionnelles et pré-entraîné sur une grande base de données telle que ImageNet et qui contiennent beaucoup de classes différentes. C'est pour cela que le modèle contient une hiérarchie robuste de caractéristiques, qui sont indépendantes de la rotation, de la translation ou encore d'une déformation spatiale.

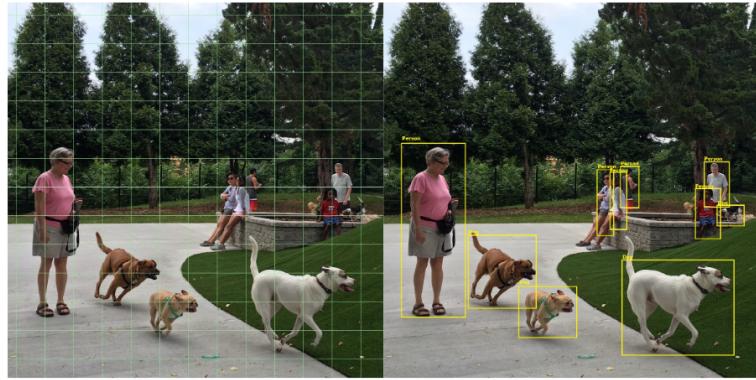


**FIGURE 5:** Modèle du VGG-16 [6]

5. J'ai voulu ensuite voir quel type d'influence aurait la base de données augmentée avec le modèle pré-entraîné VGG-16.
6. Les premières couches du CNN étant pré-entraîné à l'aide du modèle VGG-16, on va mettre à jour les poids des dernières couches à l'aide d'un réglage fin, c'est-à-dire en utilisant la rétro-propagation seulement sur les dernières couches.

## 2.2 You Only Look Once

Le principe de cette technique [7], comme son nom l'indique, est de ne regarder l'image qu'une seule fois. En effet, l'algorithme divise en  $K \times K$  cellules l'image en entrée. Il comporte une seule couche de CNN. Le but de l'algorithme est de déterminer les bounding boxes des objets présents sur une image en même temps que leur classe avec le CNN. Le tout n'étant effectué qu'une fois.



**FIGURE 6:** Division de l'image par l'algorithme YOLO et sa sortie

C'est un des algorithmes les plus rapides pour la reconnaissance d'objet dans une image. En effet, la plus récente version de cet algorithme peut aller jusqu'à 150 images par secondes. Avoir directement l'image en entrée crée des informations contextuelles qui facilitent la reconnaissance de la classe d'un objet présent sur l'image. Cela permet aussi d'éviter d'avoir l'arrière-plan qui est perçu comme un objet. De plus, quand le réseau est entraîné, il apprend plus facilement à généraliser des objets. Cela rend la détection de nouveau domaine d'objet plus facile. Mais le principal souci de cet algorithme est sa précision, surtout pour les petits objets. En effet, la bounding box qui comprend de petits objets peut être perçue comme une erreur par l'algorithme. Cet algorithme est le plus souvent utilisé avec la base de données Pascal VOC, avec laquelle je devais initialement travailler. Mais j'ai décidé d'utiliser la base de données Chien et Chat afin de pouvoir faire une étude expérimentale qualitative et comparative entre les deux algorithmes.

### 3 Méthodologie expérimentale

J'ai donc fait plusieurs études expérimentales comme mentionnées dans la partie 2 de ce document. Tous les programmes que j'ai utilisés sont sur mon [GitHub](#).

#### 3.1 Description des bases de données

J'ai utilisé deux bases de données durant mes expérimentations :

- **La base de données classique :** La base de données que j'ai utilisés pour mon étude expérimentale est la base de donnée Chats et Chiens (Dogs and Cats). Je voulais initialement travailler avec la base de données Pascal VOC mais j'ai trouvé plus intéressant de faire plusieurs expérimentations et ainsi élargir ma compréhension des systèmes de réseaux de neurones plutôt que travailler avec une plus grosse base de données et de ne pas pouvoir essayer les différentes expérimentations.  
En effet, la base de données Chats et Chiens est retrouvé sur le site [Kaggle](#). Elle contient 25000 photos dont 12500 de chats et 12500 de chiens.
- **La base de données augmentée :** L'augmentation de données est une technique qui répond à deux problématiques liées : répondre au manque de données et éviter l'overfitting d'un modèle. Son efficacité était prouvée dans des papiers tels que le très bon de Jason Wang et al. [?]. Pour m'approcher un peu plus d'une base de données telle que Pascal VOC, et avoir moins d'overfitting pour les résultats (voir partie résultats), j'ai généré 21 nouvelles images complètement aléatoires à partir d'image existante à l'aide de la fonction `ImageDataGenerator` de [Keras](#). J'ai donc 22 images à la place d'une image. Cette fonction prend en entrée plusieurs paramètres tels qu'un facteur de zoom, un intervalle de rotation, une translation 2D et un effet miroir.

#### 3.2 Protocole d'évaluation et mesures de performances (qualités et ressources)

J'ai travaillé sur l'environnement de PyCharm, un IDE permettant de coder facilement en python. Les différentes librairies que j'ai utilisées pour mes algorithmes sont : Darknet, Keras, Tensorflow-gpu, Tensorflow, Matplotlib, Sklearn, Pillow, pandas et numpy.

##### 3.2.1 CNN

Afin d'évaluer mon étude expérimentale, j'ai réduit et divisé ma base de données en 3 pour pouvoir avoir une base d'entraînement, une base de validation et une base de test. La base d'entraînement est composée de 3000 images de chats et de chiens (1500 chacun). Celle de la validation est composée de 1000 images de chats et de chiens (500 chacun). Enfin, la base de validation est composée elle aussi de 1000 images de chats et de chiens (500 chacun). Ceci dans le but comme dit précédemment d'avoir une contrainte de taille de base de données et de temps afin d'étudier tout les modèles.

Avec un batch size de 30, c'est-à-dire le nombre total d'images données aux modèles par itérations, et ayant un total de 3000 images, j'ai donc entraîné le réseau sur 30 epochs avec 100 itérations chacune.

Les valeurs que j'ai donc pu mesurer pour les différents modèles sont les courbes de précision et de coût (loss, donc la différence entre ce que l'on veut et ce que l'on a) du modèle en entraînement et en validation. Pour le test, j'ai récupéré les valeurs de précision du système (Est-ce que les prédictions sont identiques?), d'accuracy (le système est-il correct pour identifier les chats et les chiens?), le recall, permettant d'avoir une information sur la sensibilité du système et enfin le score F1, moyenne harmonique entre la précision et le recall.

### 3.2.2 YOLO

Pour le système YOLO, j'ai voulu étudier l'accuracy et le temps de calcul total de l'algorithme sur la base de données Chats et Chiens. J'ai pu récupérer sur [GitHub](#) l'algorithme YOLO, exécutable sur Windows, et utilisant Keras plutôt que Darknet. En effet, j'ai eu des soucis sur ma machine qui m'ont dirigé vers Keras plutôt que la librairie officielle de YOLO, Darknet.

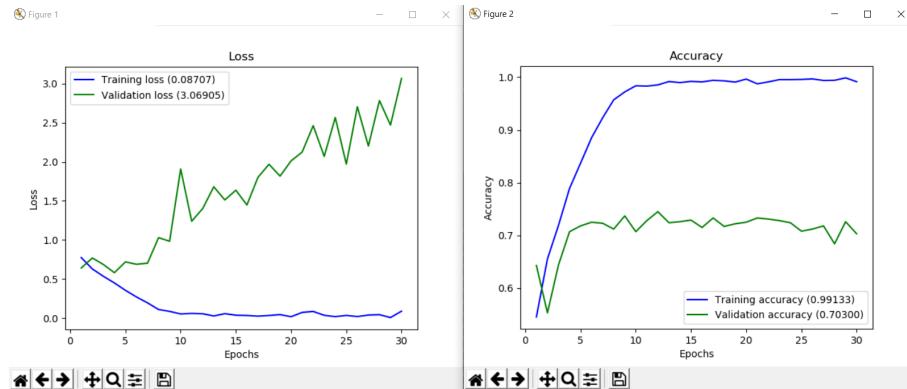
## 4 Résultats de simulations

### 4.1 Représentation des résultats

#### 4.1.1 CNN

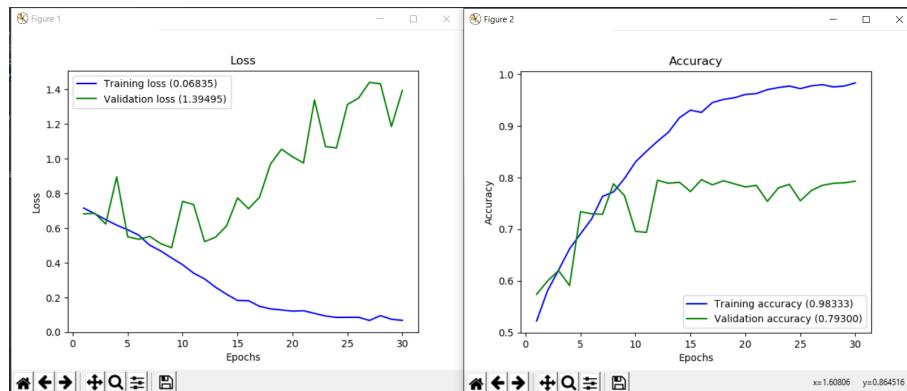
Comme dit précédemment, j'ai donc récupéré les courbes de coût et d'accuracy de la phase d'entraînement et de validation. Pour la phase de test, j'ai récupéré l'accuracy, la précision, le recall et le score F1.

1. Pour le CNN "from scratch", voici les courbes obtenus :



**FIGURE 7:** Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenus pour le CNN

2. Pour le CNN avec la régularisation, voici les résultats obtenus :



**FIGURE 8:** Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec régularisation

Model Classification report:				
	precision	recall	f1-score	support
cat	0.74	0.79	0.77	500
dog	0.77	0.73	0.75	500
micro avg	0.76	0.76	0.76	1000
macro avg	0.76	0.76	0.76	1000
weighted avg	0.76	0.76	0.76	1000

**FIGURE 9:** Rapport de classification du modèle CNN avec régularisation

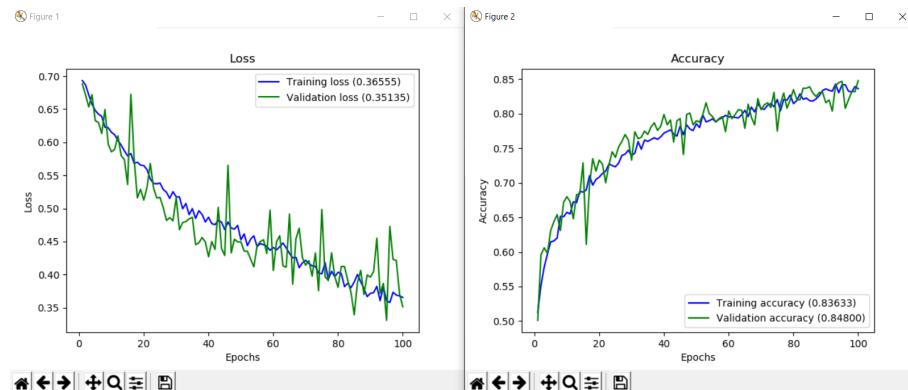
```
Model Performance metrics:
-----
Accuracy: 0.759
Precision: 0.7599
Recall: 0.759
F1 Score: 0.7588
```

**FIGURE 10:** Métrique obtenue pour le CNN avec régularisation

```
Prediction Confusion Matrix:
-----
Predicted:
      cat   dog
Actual: cat    394  106
        dog   135  365
```

**FIGURE 11:** Matrice de confusion obtenue pour le CNN avec régularisation

3. Pour le CNN avec l'augmentation aléatoire de la base de données, les résultats obtenus sont les suivants :

**FIGURE 12:** Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec augmentation de données

Model Classification report:				
	precision	recall	f1-score	support
dog	0.83	0.89	0.86	500
cat	0.88	0.82	0.85	500
micro avg	0.85	0.85	0.85	1000
macro avg	0.86	0.85	0.85	1000
weighted avg	0.86	0.85	0.85	1000

**FIGURE 13:** Rapport de classification du modèle CNN avec augmentation de données

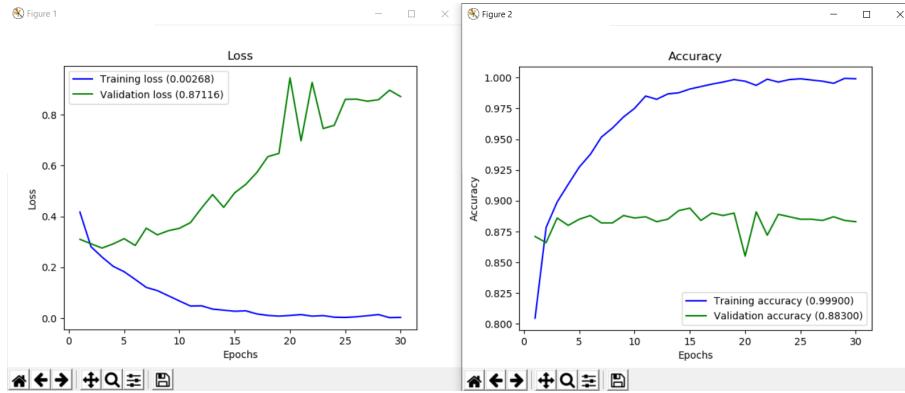
```
Model Performance metrics:
-----
Accuracy: 0.854
Precision: 0.8556
Recall: 0.854
F1 Score: 0.8538
```

**FIGURE 14:** Métrique obtenue pour le CNN avec augmentation de données

Prediction Confusion Matrix:		
Predicted:		
	dog	cat
Actual: dog	444	56
cat	90	410

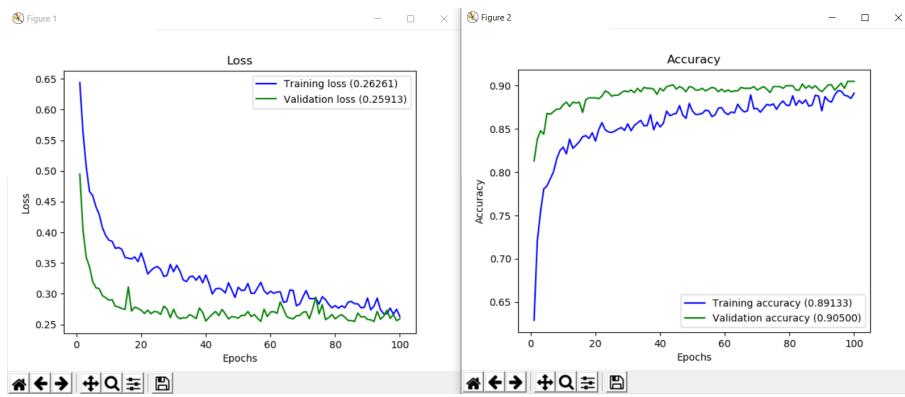
**FIGURE 15:** Matrice de confusion obtenue pour le CNN avec augmentation de données

4. Pour l'utilisation du modèle VGG-16, les résultats sont les suivants :



**FIGURE 16:** Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec le VGG

5. Pour l'utilisation du modèle VGG-16 avec l'augmentation de données, les résultats obtenus sont les suivants :



**FIGURE 17:** Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec le VGG-16 et l'augmentation de données

Model Classification report:				
cat	0.88	0.92	0.90	500
dog	0.91	0.88	0.89	500
micro avg	0.90	0.90	0.90	1000
macro avg	0.90	0.90	0.90	1000
weighted avg	0.90	0.90	0.90	1000

**FIGURE 18:** Rapport de classification du modèle CNN avec le VGG-16 et l'augmentation de données

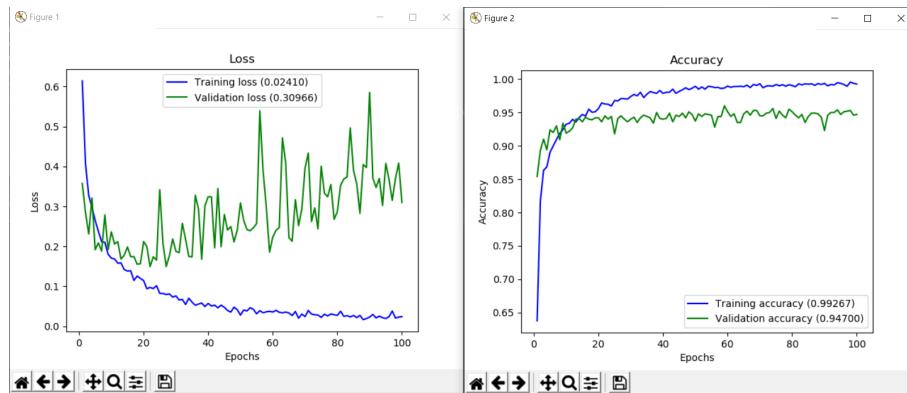
```
Model Performance metrics:
-----
Accuracy: 0.896
Precision: 0.8966
Recall: 0.896
F1 Score: 0.896
```

**FIGURE 19:** Métrique obtenue pour le CNN avec le VGG-16 et l'augmentation de données

```
Prediction Confusion Matrix:
-----
Predicted:
          cat   dog
Actual: cat    458   42
        dog     62  438
Predicted=cat
```

**FIGURE 20:** Matrice de confusion obtenue pour le CNN avec le VGG-16 et l'augmentation de données

- Pour l'utilisation du modèle VGG-16 avec l'augmentation de données et le réglage fin des dernières couches, les résultats obtenus sont les suivants :



**FIGURE 21:** Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec le VGG-16, l'augmentation de données et le réglage fin

Model Classification report:				
	precision	recall	f1-score	support
dog	0.95	0.97	0.96	500
cat	0.97	0.95	0.96	500
micro avg	0.96	0.96	0.96	1000
macro avg	0.96	0.96	0.96	1000
weighted avg	0.96	0.96	0.96	1000

**FIGURE 22:** Rapport de classification du modèle CNN avec le VGG-16, l'augmentation de données et le réglage fin

```
Model Performance metrics:
-----
Accuracy: 0.962
Precision: 0.9622
Recall: 0.962
F1 Score: 0.962
```

**FIGURE 23:** Métrique obtenue pour le CNN avec le VGG-16, l'augmentation de données et le réglage fin

```
Prediction Confusion Matrix:
-----
Predicted:
          dog  cat
Actual: dog    486  14
         cat     24  476
```

**FIGURE 24:** Matrice de confusion obtenue pour le CNN avec le VGG-16, l'augmentation de données et le réglage fin

Pour le temps d'exécution, l'algorithme le plus lent était le modèle du CNN avec l'augmentation de données de la database et qui était d'environ 7400s, soit environ 2h de temps d'exécution. Avec le VGG-16, on s'aperçoit que le temps est le même.

#### 4.1.2 YOLO

Pour le YOLO, j'ai pu récupérer le temps d'exécution total du programme qui est de 4043.56, soit 1h07. L'accuracy pour la classe chien est de 0.93% et celle de la classe chat est de 0.91%. L'accuracy du système est de 0.92%

De plus, en termes de précision, sur les 12500 images de chat du dataset données en entrée, il a réussi à en reconnaître que 11911. Il a donc reconnu 589 chiens au lieu de chats. Le système est précis à 95.3% pour les chats.

Enfin, YOLO a su reconnaître 11766 chiens sur les 12500, il a donc reconnu 734 chats au lieu de chien. La précision pour les chiens est de 94.1%.

La précision totale du système est de 94.7%.

## 4.2 Interprétation des résultats

### 4.2.1 CNN

Pour un modèle de CNN simple, sans régularisation, on s'aperçoit très rapidement que le système converge vers un taux de validation de 70%. Il s'agit en fait d'overfitting, dû à la base de données très petite. L'overfitting est dû au manque de données d'entraînement, le système voit toujours les mêmes images et il n'arrive pas à classer les nouvelles images qu'il

rencontre. Pour améliorer le système, on rajoute la régularisation. On s'aperçoit converge un taux de validation beaucoup plus grand, 79%. Mais le problème d'overfitting est toujours présent, c'est pour cela que l'augmentation de la base de données nous donne de meilleurs résultats. Avoir plus de données empêche d'avoir de l'overfitting.

En effet, malgré les pics, on s'aperçoit qu'il n'y a plus du tout d'overfitting. Le modèle converge vers un taux de validation proche de celui de l'entraînement.

Au niveau des coûts des modèles, on interprète les résultats de la façon suivante : plus il y a de données, et plus le modèle peut être généralisé et utilisé pour de nouvelles images.

Pour la partie CNN avec VGG-16, l'accuracy est bien meilleure que celle du CNN, autour de 89%. Mais on remarque toujours la présence de l'overfitting, qui est dû à la petite base de données envoyée en entrée.

Une fois qu'on lui envoie la base de données augmentée, on s'aperçoit de l'absence d'overfitting est d'une bien meilleure accuracy, 90%. Le modèle est nettement meilleur que les précédents car il est aussi beaucoup plus stable (absence de pic).

Avec un réglage fin, on obtient une accuracy de 94% pour la phase de validation.

Avec un modèle pré-entraîné, le système a donc gagné environ 25% d'accuracy. Cela montre bien l'utilité des modèles pré-entraînés pour les CNN et leurs performances.

Pour la phase de test, on observe exactement les mêmes résultats. De loin, le modèle VGG-16 avec l'augmentation de données et le réglage fin est le modèle qui nous donne les meilleurs résultats. Le CNN simple est celui qui nous donne les pires.

#### 4.2.2 YOLO

L'algorithme YOLO est très connu pour sa vitesse d'exécution. Sur une base de données comme chien et chat, il s'exécute en 1h, donc beaucoup plus rapidement que le modèle de CNN simple étudié. Malgré sa rapidité, j'ai pu avoir de bons résultats, notamment avec une accuracy de 0.92% dans la phase de test. Je n'ai pas effectué d'entraînement ni de validation car il l'était avec la base de données COCO. Je voulais juste faire la phase de test afin d'évaluer de façon plus efficace l'algorithme. Concernant la précision, j'ai des résultats très proches du VGG-16 avec l'augmentation de données et le réglage fin, 0.94%. L'algorithme est donc fidèle au même réflexion faite pour l'interprétation des modèles de CNN. Il est lui-même constitué d'une couche de CNN, c'est donc tout à fait normal de retomber sur des résultats très proches pour une petite base de données. Je pense que les résultats seraient néanmoins différents pour des bases de données telles que Pascal VOC.

## 5 Conclusion

Mon projet initial a divergé de l'étude des réseaux de neurones convolutifs à travers l'algorithme You Look Only Once vers l'étude des réseaux de neurones convolutifs à travers différentes étapes à l'aide d'une base de données restreinte. J'ai pu obtenir de très bons résultats présentés dans la partie Résultats. J'ai réussi à faire la reconnaissance des espèces animales présentes sur une image à travers mes deux types d'expérimentation :

Pour conclure sur mes résultats, YOLO a effectivement un temps d'exécution très restreint. Il est beaucoup plus rapide que le meilleur modèle que j'ai pu obtenir avec le CNN, le VGG-16, l'augmentation de données et le réglage fin pour les mêmes résultats finaux.

Pour mes recommandations futures, je pense que reprendre la base de données Pascal VOC à travers les expérimentations que j'ai menée et sur l'algorithme YOLO permettrait d'affiner les résultats que j'ai obtenus et de faire une comparaison sur une plus grande base de données.



**FIGURE 25:** Résultats obtenus par les deux algorithmes

## Table des figures

1	Photo de différentes espèces d'animaux dans des milieux environnementaux variés [1] . . . . .	3
2	Constitution des clusters neuronaux chez l'humain [4] . . . . .	5
3	Illustration d'une architecture typique d'un réseau neuronal convolutif [5] . . . . .	5
4	Résultat issu de la génération aléatoire d'image à partir de celle d'un chat . . . . .	7
5	Modèle du VGG-16 [6] . . . . .	7
6	Division de l'image par l'algorithme YOLO et sa sortie . . . . .	8
7	Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenus pour le CNN . . . . .	11
8	Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec régularisation . . . . .	11
9	Rapport de classification du modèle CNN avec régularisation . . . . .	12
10	Métrique obtenue pour le CNN avec régularisation . . . . .	12
11	Matrice de confusion obtenue pour le CNN avec régularisation . . . . .	12
12	Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec augmentation de données . . . . .	12
13	Rapport de classification du modèle CNN avec augmentation de données . . . . .	12
14	Métrique obtenue pour le CNN avec augmentation de données . . . . .	13
15	Matrice de confusion obtenue pour le CNN avec augmentation de données . . . . .	13
16	Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec le VGG . . . . .	13
17	Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec le VGG-16 et l'augmentation de données . . . . .	13
18	Rapport de classification du modèle CNN avec le VGG-16 et l'augmentation de données . . . . .	14
19	Métrique obtenue pour le CNN avec le VGG-16 et l'augmentation de données . . . . .	14
20	Matrice de confusion obtenue pour le CNN avec le VGG-16 et l'augmentation de données . . . . .	14
21	Courbes d'accuracy et de coût pour la phase de validation et d'entraînement obtenues pour le CNN avec le VGG-16, l'augmentation de données et le réglage fin . . . . .	14
22	Rapport de classification du modèle CNN avec le VGG-16, l'augmentation de données et le réglage fin . . . . .	15
23	Métrique obtenue pour le CNN avec le VGG-16, l'augmentation de données et le réglage fin . . . . .	15
24	Matrice de confusion obtenue pour le CNN avec le VGG-16, l'augmentation de données et le réglage fin . . . . .	15
25	Résultats obtenus par les deux algorithmes . . . . .	17

## Références

- [1] Ours polaire sur une banquise. <https://www.flickr.com/photos/marthaenpiet/2890352130/>. Accessed : 01-30-2019.
- [2] Alexander Gomez, Augusto Salazar, and Francisco Vargas. Towards automatic wild animal monitoring : Identification of animal species in camera-trap images using very deep convolutional neural networks. *arXiv preprint arXiv:1603.06169*, 2016.
- [3] Guobin Chen, Tony X Han, Zhihai He, Roland Kays, and Tavis Forrester. Deep convolutional neural network based species recognition for wild animal monitoring. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 858–862. IEEE, 2014.
- [4] Neuroscience : How do synapses form ? <https://www.quora.com/Neuroscience-How-do-synapses-form>. Accessed : 03-06-2019.
- [5] Hung Nguyen, Sarah J MacLagan, Tu Dinh Nguyen, Thin Nguyen, Paul Flemons, Kylie Andrews, Euan G Ritchie, and Dinh Phung. Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 40–49. IEEE, 2017.
- [6] A comprehensive hands-on guide to transfer learning with real-world applications in deep learning. <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>. Accessed : 10-04-2019.
- [7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.