Université du Québec à Montréal

Rapport Final du Travail de Recherche
INF7370 - Apprentissage automatique

Implémentation d'un modèle d'apprentissage profond pour la distinction entre chats et chiens et un transfert d'apprentissage vers la distinction entre chiens et coyotes

Nom complet de l'étudiant : Maroun Haddad
Code permanaent : HADM15088902

Nom complet de l'étudiant : Mohamed Fawzi Touati
Code permanent : TOUM13129401

29 Avril 2019

Sommaire

Table des figures

Liste des tableaux

Introduction générale

L'apprentissage automatique est un processus qui permet aux machines d'apprendre d'un échantillon d'exemples fournit soit de manière supervisée ou non supervisé. L'apprentissage profond fait une étape de plus que l'apprentissage automatique traditionnelle car il contribue à faire des pré traitements qui sont propres à lui (exp : phases de convolution et échantillonnage).

La classification des images est le processus consistant à prendre une entrée (comme une image) et à sortir une classe (comme «Objet») ou une probabilité que l'entrée soit une classe particulière («il existe une probabilité de 90% que cette entrée soit un objet en particulier»). En général, les hommes peuvent regarder une photo et savoir à quel objet elle appartient, mais comment un ordinateur peut-il apprendre à le faire ?

Dans le cas d'un apprentissage machine qui fait la distinction entre différents types d'objets, la capacité de généralisation et de prédiction est un critère important puisque de nombreuses applications dans les domaines des technologies, des transports, de l'environnement et de la recherches en dépendent.

Dans le travail de recherche du cours INF 7370- Apprentissage automatique, il sera question de traiter une problématique en apprentissage profond et de développer un modèle de classification des images des chiens, chats et coyotes dans le but de faire la distinction entre ces types d'animaux.

Le choix du sujet a été fait de manière réfléchi, même si au début nous avions été pris par une confusion puisque le dataset décrit dans plusieurs travaux de l'état de l'art n'était pas suffisant afin d'effectuer une bonne distinction entre les chiens et les coyotes. Donc, dans ce qui suit, nous allons montrer toutes les solutions implémentées et suivies afin de réaliser un modèle d'apprentissage profond capable de distinguer des coyotes, des chiens et des chats en utilisant plusieurs techniques vues et notés dans ce cours et évoqués dans l'état de l'art.

Afin de réaliser cela, nous allons présenter une vue globale de notre travail qui comporte : la problématique générale, la motivation de ce projet, Une description et comparaison des travaux connexes, la présentation de notre environnement de travail et une description complète du travail à réaliser ainsi que les résultats conséquents.

1

Contexte générale et problématique

1.1 Introduction

La toute première étape de chaque travail de recherche réside dans la mise en contexte du travail de recherche. Après avoir effectué une introduction et une mise en contexte du sujet choisi, nous allons nous attarder sur la problématique principale.

1.2 Contexte générale

La distinction entre les différents objets demeure aujourd'hui une priorité, dans un monde de plus en plus connecté où les besoins ne cessent de croitre. L'un des domaines de cette distinction réside dans le domaine de l'environnement où les spécialistes doivent fournir des rapports en se basant sur des données de détection des animaux dans la nature quelle soit urbaine ou non urbaine.

Dans un contexte normal, la distribution de chats, chiens et coyotes dans un environnement quelconque est non balancée, car on remarque parfois la présence de chiens errants qui représentent un danger pour la population et pour les animaux, comme dans d'autres cas, il peut y avoir une surpopulation de chats comme c'est le cas pour certains pays ou provinces.

C'est ainsi que les chercheurs mettent des détecteurs (caméras) pour la détection de ces animaux. Pour cela, il est nécessaire pour les chercheurs en apprentissages automatique et profond, de fournir des solutions adéquates pour faire une distinction automatique des différentes races d'animaux (Chat, chiens et coyotes dans notre cas).

Les algorithmes de distinction différent, il existe des algorithmes de distinction qui sont supervisés, d'autres non, mais la disponibilité des données pour faire ces traitements demeure une étape importante puisqu'ils sont déterminants quant à l'exactitude, la performance et les résultats d'algorithmes en particulier.

1.3 Motivations

La motivation derrière ce travail réside dans le fait que les travaux de recherche dans le développement de solutions d'apprentissage profond ne traitent quasiment pas le problème de distinction entre chiens et coyotes. En effet, dans certains travaux on retrouve que les données de représentation des faces des coyotes sont très minimes comparées à ceux des chiens ou des chats. Il en existe un dataset (ensemble de données) qui représente les données des coyotes, mais uniquement avec 150 faces. Dans ce travail, nous allons mettre toutes les connaissances acquises lors de cette session et en s'appuyant sur la documentation de recherche en ligne pour implémenter une solution tout en traitant avec les données présentes.

1.4 Problématique

La distinction des images de chiens et de chats. C'est facile pour les humains, mais des preuves suggère que les chiens et les chats sont particulièrement difficiles à distinguer automatiquement (Elson *et al.*, 2007). C'est de ce contexte-là que part notre problématique principale qui est la mise en place d'un modèle de distinction entre chats et chiens pour contribuer aux travaux dans les domaines de l'environnement et de la santé. En plus, l'architecture conçue pour ce système peut être généralisée pour distinguer d'autres types d'entités ou d'animaux. La première problématique technique dans ce problème réside dans la capacité des machines à apprendre la distinction entre différents types d'animaux en l'occurrence dans ce cas les chats et chiens en utilisant des architectures différentes d'apprentissage profond ainsi que les chiens et les coyotes. La deuxième problématique technique à mettre en place, c'est la réalisation d'une solution de distinction entre chiens et coyotes en réalisant un transfert d'apprentissage de la solution des chats et des chiens vers une solution entre chiens et coyotes. Le transfert d'apprentissage réalisé avec succès dans d'autres domaines tels que la bio-informatique et la robotique, il est très difficile de construire une grande échelle bien annotée ensemble de données en raison des frais d'acquisition de données et d'annotation coûteuse, ce qui limite son développement (Tan *et al.*, 2018).

La dernière problématique représente le paramétrage des modèles de prédiction, car il faut adapter les paramètres des architectures aux paramètres propres du problème. Les données qui seront utilisées sont aussi un rempart quant à la bonne capacité d'apprentissage du modèle de prédiction puisque la distinction repose sur différentes faces de prise d'images et de la manière de prise. Dans un tel cas, prendre un dataset nettoyé et issu d'une étude préalable sur les différences morphologiques animales. La mise en place ainsi que le développement d'une technique qui propose une solution quant aux problématiques techniques et de données représentent notre défi principal dans ce travail de recherche. Enfin, dans ce projet, nous souhaitons également résoudre ce problème et obtenir de meilleures performances.

1.5 Conclusion

À la fin de ce chapitre, nous avons présenté le contexte général de la recherche, les motivations menant vers ce sujet et les problématiques reliés a ce projet.

2

État de l'art

2.1 Introduction

Après avoir explicité la problématique de ce travail de recherche dans le chapitre précédent, nous allons maintenant évoquer les principaux travaux réalisés et faire une étude comparative à ce sujet et sur lesquels nous allons nous inspirer.

2.2 Travaux connexes et études comparatives

Dans cette section, nous allons aborder une étude sur les différents travaux déjà existants pour la distinction entre les chats et les chiens, entre les chiens et les coyotes et introduire certains travaux du transfert d'apprentissage.

2.2.1 Travaux de distinction entre chats et chiens

De nombreux de travaux de distinction ont été réalisé pour cette problématique, le premier est celui de (Bang Liu, 2014) qui a fait une étude approfondie du problème de classification d'images de chats et de chiens. Dans cet article, les auteurs ont utilisé les données récoltées par *Microsoft* (Research, 2012) dans le cadre du projet *Assira*. Les auteurs ont fait une étude approfondie dans la classification d'images et ont présenté de multiples solutions en fixant plusieurs paramètres.

2.2.2 Travaux de distinction entre chiens et coyotes

En effet, selon citation scientifique (Stanford, 2018) , les chiens sont généralement des animaux domestiques (vivant avec des humains), tandis que les coyotes vivent à l'état sauvage, près de sites naturels. En second lieu, les coyotes sont plus élancés que les chiens, avec un museau plus pointu et un front plus plat, ajoutons a cela que ,la poitrine d'un chien apparaît plus profonde que celle d'un coyote, ce qui donne l'impression qu'un coyote a les pattes plus longues qu'un chien ; et enfin, les coyotes ont des pistes plus allongées que les chiens. Les travaux de distinction entre chiens et coyotes sont très peu nombreux, puisque ce sujet est très mal fourni en données permettant d'exécuter un algorithme d'apprentissage et de prédiction. Certes, les chercheurs ne s'intéressaient pas à ce sujet à cause des environnements différents dont ils sont issus, mais la problématique reste présente puisqu'il est important de distinguer ces deux races.

2.2.3 Travaux d'apprentissage par transfert

La capacité d'un système à reconnaître et à appliquer les connaissances et compétences acquises lors de précédents domaines ou tâches à de nouvelles tâches ou domaines, qui partagent quelques points communs. L'apprentissage par transfert est une des techniques utilisées dans l'apprentissage profond pour qui les problèmes de données manquantes sont très soulevés. Dans la majorité des travaux, le modèle développé pour une tâche donnée est réutilisé comme point de départ d'un modèle pour une seconde tâche. Il s'agit d'une approche populaire en apprentissage profond qui utilise des modèles préentrainés comme point de départ des tâches de traitement de la vision par ordinateur, en raison des vastes ressources de

calcul et de temps nécessaires pour développer des modèles de réseau de neurones sur ces problèmes (Pan et Yang, 2010).

Méthodes d'utilisations

Il existe deux méthodes principales d'utilisation de l'apprentissage par transfert (Yosinski *et al.*, 2014) :

- **Développer une approche modèle :**
  Quatre étapes se présentent dans ce cas :
  - Sélection d'une tache source : Le problème de modélisation prédictif doit être associé à une abondance de données pour lesquelles il existe une relation entre les données d'entrée, les données de sortie et/ou les concepts appris au cours du mappage des données d'entrée aux données de sortie.
  - Développement d'un modèle source : Dans cette étape, il va falloir développer un modèle habile pour cette première tâche. Le modèle doit être meilleur qu'un modèle naïf pour garantir que certaines fonctionnalités ont été apprises.
  - Modèle de réutilisation : L'ajustement du modèle sur la tâche source peut ensuite être utilisé comme point de départ pour un modèle sur la deuxième tâche d'intérêt. Cela peut impliquer l'utilisation de tout ou partie du modèle, en fonction de la technique de modélisation utilisée.
  - Modèle d'accord : Le modèle peut éventuellement nécessiter une adaptation ou une précision sur les données de la paire entrée-sortie disponible pour la tâche d'intérêt.
- **Utilisation d'un modèle préentrainé :**
  - Sélection du modèle source : Un modèle source préentrainé est choisi parmi les modèles disponibles. De nombreux instituts de recherche publient des modèles sur des ensembles de données volumineux et complexes qui peuvent être inclus dans les séries de modèles candidats.
  - Modèle de réutilisation : Le modèle préentrainé peut ensuite être utilisé comme point de départ pour un modèle sur la deuxième tâche d'intérêt. Cela peut impliquer l'utilisation de tout ou partie du modèle, en fonction de la technique de modélisation utilisée.
  - Modèle d'accord : Le modèle peut éventuellement nécessiter une adaptation ou une précision sur les données de la paire entrée-sortie disponible pour la tâche d'intérêt. Ce deuxième type d'apprentissage par transfert est commun dans le domaine de l'apprentissage profond.

Transfert d'apprentissage pour la classification d'images en apprentissage profond

Il est courant de réaliser un apprentissage par transfert avec des problèmes de modélisation prédictive utilisant des données d'image en entrée. Il peut s'agir d'une tâche de prédiction prenant des photos ou des données vidéo en entrée. Pour ces types de problèmes, il est courant d'utiliser un modèle d'apprentissage profond préentrainé pour une tâche de classification d'images de grande taille et difficile à distinguer (exemple : chiens et coyotes) (Yosinski *et al.*, 2014). Quelques modèles préentrainés ont été cités dans plusieurs travaux, on cite parmi ceux-là :

- Le modèle Oxford VGG.
- Le modèle Google Inception.
- Le modéle Modèle Microsoft ResNet.

Cette approche est efficace, car les images ont été formées sur un grand corpus de photographies et obligent le modèle à faire des prédictions sur un nombre relativement grand de classes.

2.3 Conclusion

Lors de ce chapitre de l'état de l'art, nous avons fait une étude des travaux connexes existants propres à ce sujet. La première étant la distinction entre chiens et chats et la deuxième étant celle du chien et coyote et nous avons terminé par les travaux reliés à l'apprentissage par transfert.

3

Analyse personnelle

3.1 Introduction

Dans ce chapitre nous allons aborder notre analyse du sujet de recherche traité et les solutions pressenties pour cela.

3.2 Analyse du probléme

L'analyse du problème se décompose en deux parties, en effet au tout début du projet, nous avions opté pour l'implémentation d'une solution de distinction en classifiant les images des chiens et des coyotes qui demeurent une problématique réelle pour certains environnements. Mais au fur et à mesure des avancements et de la documentation faite à ce sujet, nous nous sommes rendu compte que la tache demeurée relativement difficiles puisque les données nous manquaient. Donc, nous avons préféré réaliser un modèle d'apprentissage profond pour la distinction des chiens et des chats et ensuite transférer ce modèle pour une distinction entre chiens et coyotes.

3.3 Présentation de la solution pour la distinction ds chats et chiens

3.3.1 Phase 1 : Récolte et pré-traitement des données

Dans cette phase, nous allons faire une collecte de données de l'état de l'art pour les chats et les chiens. La base donnée contenant ces informations est *Assira* collectés par *Microsoft*.

3.3.2 Phase 2 : Entrainement d'un Auto-encodeur pour l'extraction des caractéristiques des chats et chiens

Dans cette phase, nous allons mettre en place une stratégie d'extraction de caractéristiques des chiens et chats. En effet, afin de permettre cela, nous allons utiliser les auto-encodeurs. Supposons maintenant que nous n'ayons qu'un ensemble d'exemples d'apprentissage non étiquetés.

Un réseau de neurones auto-encodeur est un algorithme d'apprentissage non supervisé qui applique la rétropropagation, en définissant les valeurs cibles de manière à ce qu'elles soient égales aux entrées. L'autocodeur essaie d'apprendre une fonction qui minimise l'erreur ; en d'autres termes, il essaie d'apprendre une approximation de la fonction d'identité, afin d'obtenir une sortie similaire à l'entrée.

À travers cela , nous utiliserons les auto-encodeurs afin d'extraire les caractéristiques des images présentes des chiens et des chats. En effet, nous nous intéressons à la sortie de la partie encodeur qui représente pour nous les caractéristiques principales du problème de distinction des chats et chiens. Cette phase permet d'identifier réellement les paramètres qui affectent l'apprentissage pour faire la distinction.

Cette phase représente la base de la solution de résolution de la distinction via un classifieur qu'on va traiter dans la partie suivante. Le choix d'une architecture doit être la plus adaptée au problème pour la suite et non forcément la meilleure. Enfin, nous allons présenter les différents résultats obtenus en utilisant de multiples paramètres de l'apprentissage.

3.3.3 Phase 3 : Entrainement d'un réseau de neurones convolutionnel pour la distinction entre chiens et chats

Avant de plonger profondément dans la construction du réseau de neurones convolutionnel, il est nécessaire de maitriser quelques éléments de base concernant un problème de classification des images.

La perception d'une image par la machine est complètement différente de ce que nous voyons. En fait, ce ne sont que des nombres que les machines voient dans une image. Une valeur comprise entre 0 et 255 est attribuée à chaque pixel de l'image. Ainsi, pour que la machine puisse classifier une image, un prétraitement est nécessaire pour rechercher des motifs ou des caractéristiques distinguant une image d'une autre.

Puis, vient la phase de classification via un réseau de neurones complètement connectés qui réalise la tâche d'apprentissage au fur et à mesure en fonction des paramètres donnés en entrées et le modèle pourra ensuite être utilisé pour des prédictions. Élaboré justement pour éviter la partie des prétraitements sur l'auto-encodeur, l'extraction des caractéristiques des chiens et des chats vise a évité la parie de convolution et d'échantillonnage. En effet, nous allons transférer les résultats de l'apprentissage avant d'effectuer la classification des images. Une série d'expérimentations vont être exécutées dans l'élaboration de ce modèle principale.

Une étude comparative avec l'utilisation de différents paramètres va être utilisée dans ce cas pour permettre d'avoir de bons résultats d'apprentissage et de validation.

Il est a rappeler que l'utilisation des résultats connus de l'état de l'art, c'est à dire d'un réseau de neurones convolutionnel pour la distinction entre les deux races animales ont été réimplémenter avec des images colorées de taille [150 x 150], et cela uniquement pour des fins de comparaison.

Ainsi, à la fin de la phase d'apprentissage, le problème principal de classification d'images entre les chiens et les chats se termine.

3.3.4 Phase 4 : Utilisation d'un modèle d'apprentissage pré-entrainé

Afin d'avoir une idée sur le plus bon résultat qui puisse exister dans notre travail de distinction entre chiens et chats. Nous allons utiliser le modèle *VGG-16* préentrainé avec la base d'images *ImageNet*. Pour résumer, le résultat de cette phase sera considéré comme le modèle le plus performant qui existe. Cette phase a été rajoutée par souci de comparaison uniquement.

3.4 Présentation d'une solution pour la distinction des chiens et des coyotes

3.4.1 Phase 1 : Récolte de données et pré-traitements

Dans cette phase, nous allons exécuter une récolte d'images des coyotes du web en utilisant des outils de « scrapping ». Après cela, nous allons effectuer des prétraitements manuels sur les images récoltés en se basant sur l'étiquetage des images. D'autres phases de prétraitements viendront ensuite afin d'affiner les expérimentations comme l'augmentation de données.

3.4.2 Phase 2 : Réalisation d'un transfert d'apprentissage depuis le modèle de distinction des chiens et chats

En effet, dans cette phase finale, nous avons fait un transfert d'apprentissage en utilisant les résultats du modèle d'apprentissage profond de distinction entre chats et chiens vers un modèle de classification d'images profond pour la distinction entre chiens et coyotes. Pour cela, nous allons utiliser les images de chiens et coyotes entrées du réseau de neurones convolutionnel pour produire une prédiction de classification d'images des chiens et coyotes. Enfin, dans cette phase, nous allons aborder des paramétrages différents pour avoir des résultats et les analyser convenablement.

3.5 Conclusion

À la fin de chapitre, nous avons pu déterminer la solution ou la méthodologie qui nous permettra de résoudre cette problématique.

4

Implémentation de la solution et analyse des résultats

4.1 Introduction

Après avoir analysé les solutions possibles pour notre problématique, nous allons maintenant expliciter l'implémentation faite.

4.2 Environnement et outils de travail

4.2.1 Matériels

Le matériel utilisé consiste en 1 ordinateur personnel et un GPU décrit, ci-après :.

1. **Poste de travail**

Système d'exploitation	Windows 10 64bits
RAM	16 Go
Processeur	Intel Core i7-7700 HQ CPU @ 2.80GHz

TABLE 4.1 – Caractéristiques du poste de travail

2. Solution pour le GPU Afin d'utiliser des ressources GPU performantes, nous avons opté pour *Colaboratory*. *Colaboratory* (Collab, 2019) est un Service cloud gratuit de *Google* pour les développeurs d'intelligence artificielle. Avec Colab, nous pourrons développer gratuitement des applications d'apprentissage profond sur le GPU, les données sont alors chargées sur le drive de *Google* et utilisés ensuite pour exécuter un entrainement que ce soit pour l'auto-encodeur ou le réseau de neurones conventionnel.

4.2.2 Langages de programmation et logiciels

Nous avons utilisé au cours de la réalisation de ce projet *Python* qui est un langage de programmation de haut niveau. Il supporte la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort et d'une gestion automatique de la mémoire. Plusieurs bibliothèques sont fournies afin de faciliter les développements dans le domaine de l'apprentissage profond comme *Keras* (Keras, 2019).

4.3 Distinction entre chats et chiens

4.3.1 Préparation des données

Pour les données des chats et des chiens, nous avons utilisé les images offertes par la compétition de *Kaggle dogs-vs-cats* (Kaggle, 2013). Les données sont divisées en deux parties : 25,000 images d'apprentissage (12,500 images de chats et 12,500 images de chiens) et 12,500 images de test. Or, les images de test ne sont pas séparées en deux classes, car dans la compétition l'évaluation des modèles est effectuée par les évaluateurs du concours, donc notre seule solution été de les annoter manuellement.

Comme c'est fastidieux d'annoter 12,500 images à la main, et afin d'automatiser ce processus, , nous avons développé un logiciel *Images_Laberler* (figure[4.1]) avec le langage C# qui accélère ce processus.

Le programme charge toutes les images en mémoire et en cliquant sur le bouton gauche l'image est automatiquement copiée dans le dossier des chats et de même le bouton droit effectue la même opération pour les images des chiens. Le logiciel numérote les images séquentiellement et peut être adopté pour n'importe quelle classification binaire.

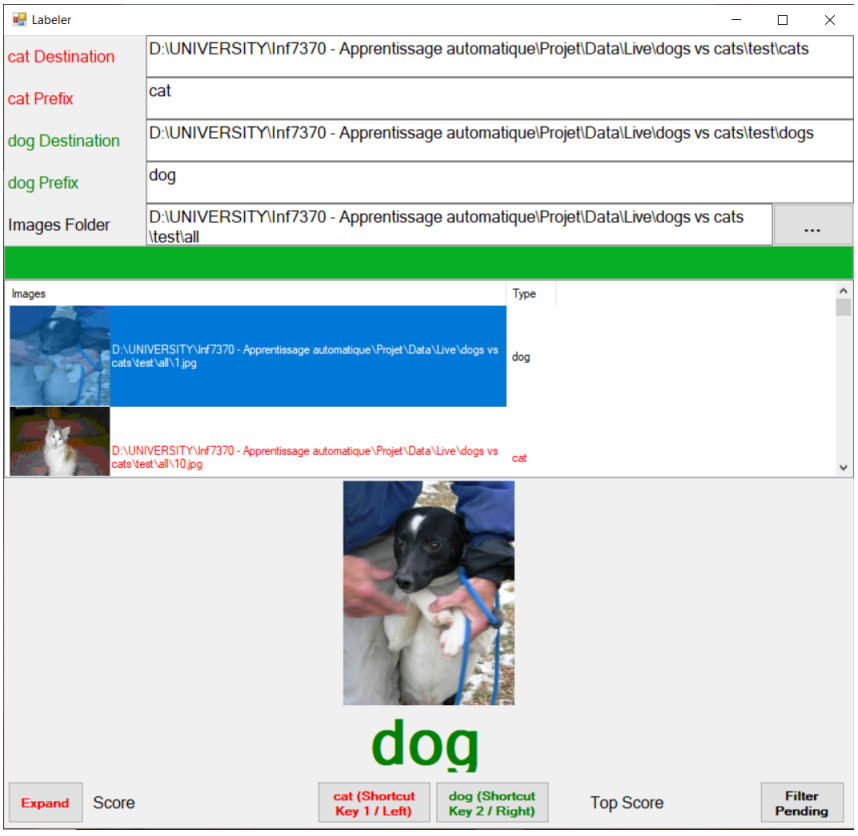

FIGURE 4.1 – Outil d'étiquetage des images

4.3.2 Auto-encodeur

Introduction

L'auto-encodeur est un réseau de neurones multicouche construit de deux parties (Encodeur et Decodeur). Le rôle de l'encodeur est de réduire l'image à ces dimensions les plus pertinentes (appelé Bottleneck) et le rôle du decodeur est de reconstruire l'image initiale à partir de ces dimensions. Il existe plusieurs types d'auto-encodeur (Auto-encodeur Multicouches, auto-encodeur Regularisé et Denoising auto-encodeur) mais pour notre étude nous utilisons les auto-encodeurs Convolutionels.

Le rôle de notre auto-encodeur est de trouver les dimensions les plus pertinentes des images des chats et des chiens afin d'accélérer et d'améliorer la performance de notre classifieur. Pour cela, l'architecture de la partie Encodeur doit être une copie exacte de celle du classifieur et les spécifications des données d'entrée (exemple couleur et taille) doivent être identiques pour le classifieur et l'auto-encodeur.

Pour évaluer la performance de nos modèles, nous utiliserons deux facteurs :

1. Mean squared error : l'Erreur quadratique moyenne. Plus cette valeur est petite, plus le modèle performe bien.
2. Qualité de la compression : La qualité des images reconstruites par le decodeur. Plus elles sont proches des images d'entrée, plus le modèle performe bien.

Or, après plusieurs expérimentations nous avons constaté que ces valeurs ne sont pas toujours fiables à représenter la puissance réelle de l'auto-encodeur à capturer les dimensions pertinentes et par conséquent à aider le classifieur. Ceci sera détaillé dans la section suivante.

Facteurs contribuant à la performance

Nous avons effectuer des expérimentations sur plusieurs modèles et architectures et par conséquent nous avons isolé les facteurs les plus pertinents qui affectent l'apprentissage de l'auto-encodeur.

1. **La taille du Bottleneck :** la taille du Bottleneck est la partie la plus importante de l'architecture. Si le Bottleneck est grand, c.-à-d. on n'a pas réduit suffisamment la taille des images par DownSampling, l'auto-encodeur va converger rapidement avec une *Erreur de validation* très petite et les images reconstruites par le decodeur vont être très proches des images d'entrées.

   Cependant, si on utilise cet auto-encodeur avec le classifieur, la performance de ce dernier va être pire. La raison étant qu'avec un grand Bottleneck, le decodeur va être capable de reconstruire facilement les images sans apprendre aucune information sur leur structure. Mais si on réduit la taille du Bottleneck, même si le decodeur va éprouver une grande difficulté à reconstruire les images initiales et la convergence va prendre plus de temps, le modèle va apprendre des caractéristiques plus pertinentes. Il est important de mentionner que la taille du Bottleneck impose la profondeur de notre modèle et elle est de même affectée par les détails des images d'entrée.

2. **Les couleurs des images d'entrée :** les images colorées offrent plus de détails pour l'apprentissage et par conséquent améliore la performance du classifieur, mais ils sont très difficiles à être reproduites par le decodeur et contribuent considérablement au ralentissement de l'apprentissage de l'auto-encodeur comme elles imposent 3 dimensions au lieu de 1 comme c'est le cas avec les images en noir et blanc. Dans notre étude on a opté à utiliser les images en noir et blanc avec l'auto-encodeur, car les images colorées demandaient beaucoup plus de ressources, ce qui n'était pas possible avec notre environnement. La figure [4.2] montre quelques résultats préliminaires que nous avons obtenus avec les images colorées, mais comme la figure [4.3] montre, nous n'avons pas pu réduire la perte de moins que 0.5 et par conséquent la performance de notre classifieur s'est détériorée quand il était fusionné avec l'auto-encodeur à images colorées.

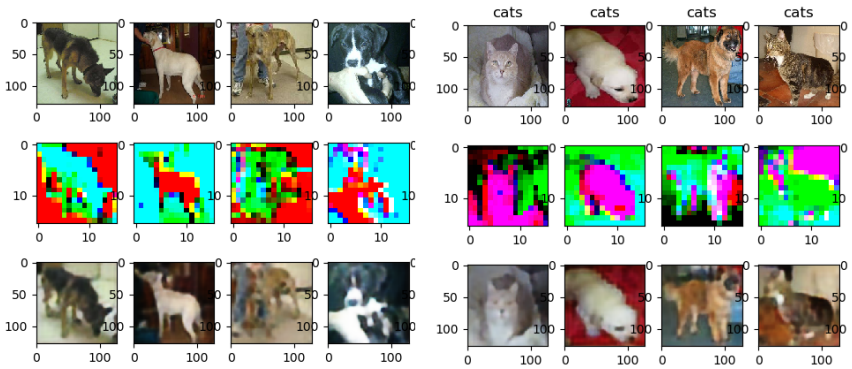

FIGURE 4.2 – Auto-encodeur Coloré - Échantillons de compression (Adadelta | Sgd)

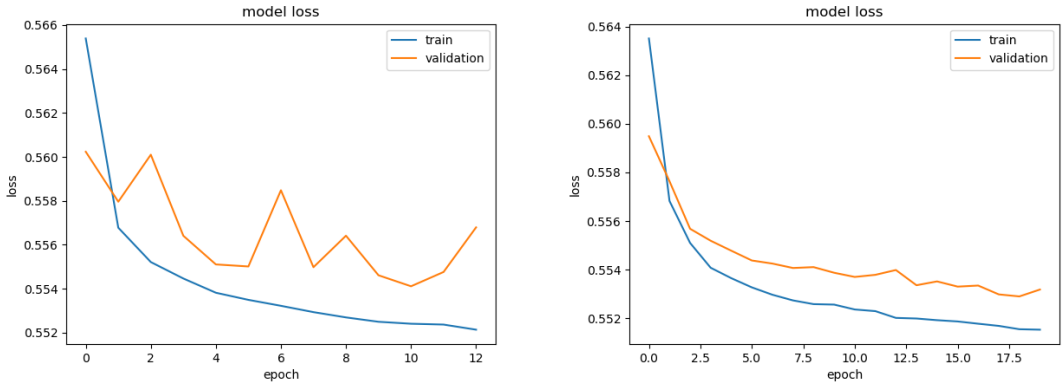

FIGURE 4.3 – Auto-encodeur Coloré - Fonction de perte (Adadelta | Sgd)

3. **La taille des images d'entrée :** Plus les images d'entrée sont grandes plus elles vont contenir des détails pour aider le classifieur. Or, la taille des images impose la profondeur de notre auto-encodeur, car pour réduire les grandes images à la taille optimale du Bottleneck on a besoin d'une architecture très profonde. Plus l'architecture est grande, plus le nombre de paramètres à entrainer augmente. Dans notre étude on a expérimenté avec les tailles suivantes pour l'auto-encodeur (128x128, 150x150 et 200 x 200), mais enfin on a opté à utiliser la taille 128x128. La raison étant que nos ressources ne sont pas suffisantes pour les images de grandes tailles si on veut entrainer sur les 25,000 images.
4. **La taille de l'échantillon d'apprentissage :** Plus l'échantillon d'apprentissage est grand plus la possibilité que le modèle converge à une petite erreur de validation va augmenter. Cependant, plus la taille de l'échantillon est grande plus la taille de la mémoire nécessaire pour charger les images va augmenter et plus la durée du temps nécessaire pour compléter chaque itération va augmenter.
5. **La profondeur de l'architecture :** le nombre de couches de l'auto-encodeur est très corrélé avec la taille du Bottleneck et par conséquent joue un rôle important dans l'apprentissage. En général, plus l'architecture est profonde, plus l'apprentissage s'améliore (Srivastava *et al.*, 2015). Cependant, plus qu'on augmente le nombre de couches, plus le nombre de paramètres à apprendre va augmenter et par conséquent plus de ressources et de temps sont nécessaires pour que le modèle converge. Nous avons aussi constaté qu'augmenter le nombre de couches n'est pas toujours positif. Enfin, on a remarqué que les architectures de l'encodeur et du decodeur sont dictées par l'architecture du classifieur et alors nous n'avons pas beaucoup de liberté avec les changements d'architecture de l'auto-encodeur.
6. **La normalisation du batch :** Durant nos expérimentations, nous avons constaté que l'ajout de l'étape de la normalisation du batch après chaque convolution a aidé notre modèle à bien reconstruire les images d'entrée. Ceci est confirmé par d'autres études (Ioffe et Szegedy, 2015).
7. **Optimiseur :** Le choix de l'optimiseur est crucial pour garantir la convergence de notre modèle. Durant notre étude nous avons essayé plusieurs optimiseurs (Adadelta, sgd et Adaboost) avec plusieurs paramètres pour le taux d'apprentissage. Nous avons opté à utiliser l'optimiser sgd avec un taux d'apprentissage de 0.1, un momentum de 0.9 et un decay de 10^{-6}. le taux d'apprentissage avec le momentum et le decay sont très important pour parvenir le modèle de tomber dans un minimum local.

Architecture finale - images en Noir et Blanc [128x128]

Après plusieurs expérimentations et en considérant le nombre de données que nous devrons traiter et la capacité de notre environnement de travail, nous avons choisi le modèle suivant comme modèle final de l'auto-encodeur (figure [4.4]). Le modèle prend des images noires et blanches de taille 128x128 avec 57 couches en total et 2,937,025 paramètres à entrainer.

L'Encodeur est composé de 4 sections (28 couches), chaque section est formée de :

» Conv > ReLu > BN > Conv > ReLu > BN > Pooling.

Les filtres des convolutions évoluent comme suite :

» Conv32>Conv32>Conv64>Conv64>Conv128>Conv128>Conv256>Conv256

La taille des images est réduite comme suite :

» 128x128 > 64x64 > 32x32 > 16x16

Alors à la sortie de l'Encodeur, le Bottleneck aura les dimensions 16x16x256 (small and thick) où 16x16 est la taille des images et 256 le nombre de filtres.

La structure du decodeur est un miroir de celle de l'Encodeur avec l'échantillonnage remplacée par UpSampling qui va agrandir les images afin qu'elles retournent à leurs tailles initiales. Comme fonction d'activation, nous avons utilisé ReLu pour les couches cachées et Sigmoïde pour la couche de sortie.

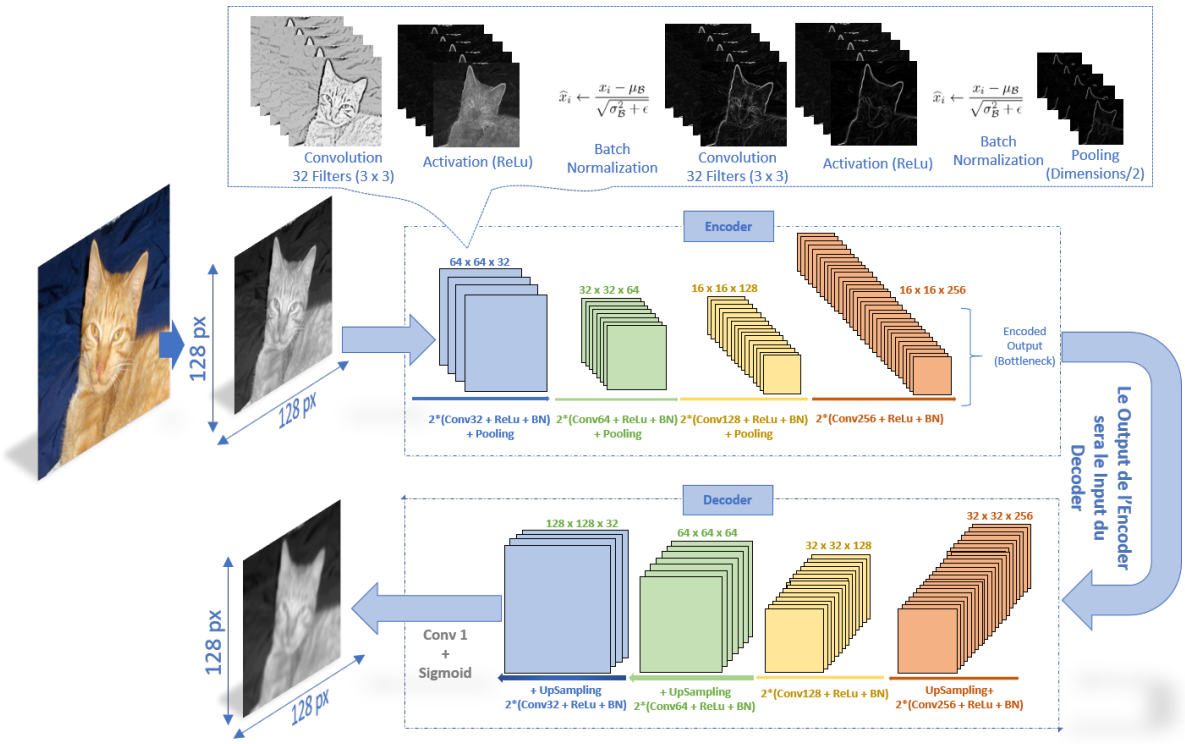

FIGURE 4.4 – Auto-encodeur - Architecture finale

Résultats de l'apprentissage

L'apprentissage de notre modèle final a pris 2 jours et 6 heures. Il est effectué sur 400 itérations avec 20,000 images d'apprentissage et 5,000 images de validation. Les données sont mélangées (les classes réelles ne jouent pas un rôle) alors que les images d'entrées elles-mêmes sont utilisées comme sortie ou classes réelles pour l'apprentissage. Les proportions des données ainsi que les paramètres utilisés sont détaillés dans les tableaux [4.2] et [4.3] respectivement.

Le modèle a convergé avec une *Validation Loss* de 0.0026 (figure [4.5]), la meilleure valeur obtenue sur l'ensemble des données total avec un Bottleneck de taille 16x16. La qualité de la compression des images est très grande comme c'est évident dans la figure [4.6].

La performance de l'auto-encodeur avec le classifieur et son effet sur ce dernier sont détaillés dans la section [**1.3 classifieur - Conclusion sur L'efficacité de l'auto-encodeur**].

	Apprentissage	**Validation**	**Total**
Chats	10,000	2,500	12,500
Chiens	10,000	2,500	12,500
			25,000

TABLE 4.2 – Auto-encodeur - Détails des données

Paramètres							
Couleur	**Largeur**	**Taille**	**Itérations**	**Patience**	**Activation**	**Param.**	**couches**
Gris	128	128	400	10	ReLu	2,937,025	57
Résultats							
Perte de validation				**Temps(H)**		**Early Stopping**	
0.0026				54 heures		No Early Stopping	

TABLE 4.3 – Auto-encodeur - Paramètres et Résultats

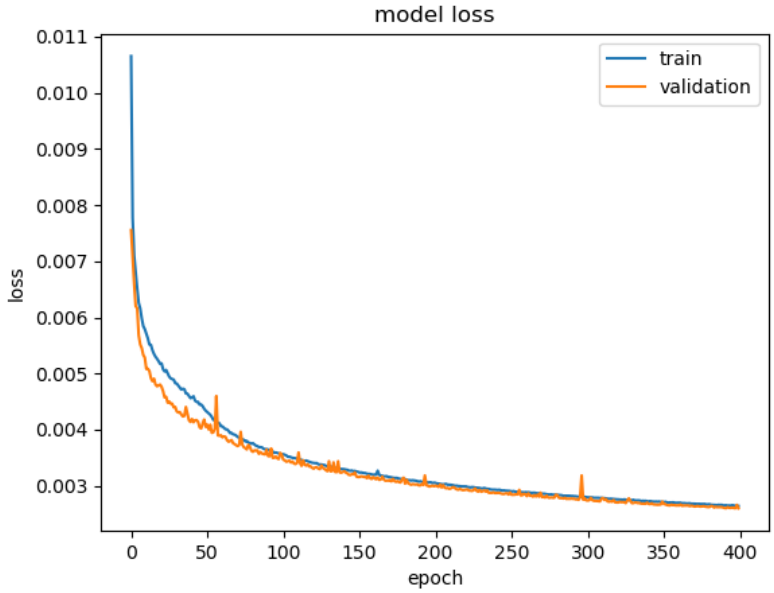

FIGURE 4.5 – Auto-encodeur - Courbe de perte

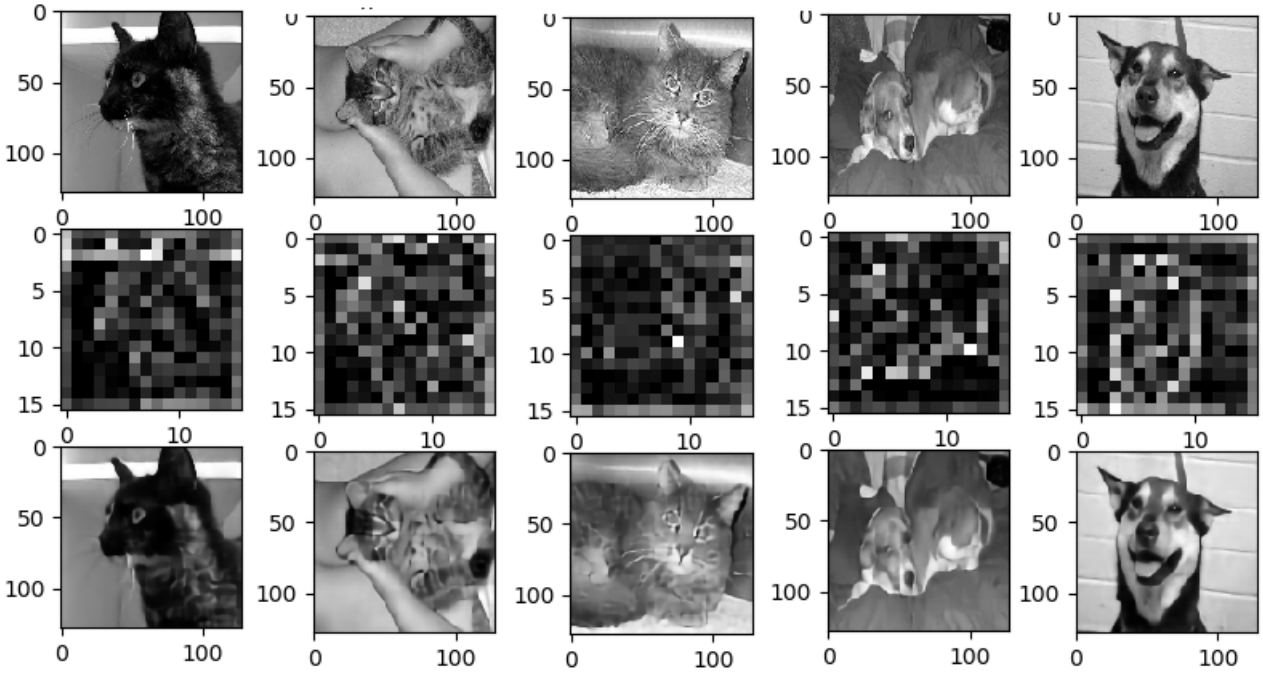

FIGURE 4.6 – Auto-encodeur - Échantillons de compression

4.3.3 Classification des images

Introduction

Pour classifier les images des chats et des chiens, nous entrainons un réseau de neurones convolutionel (CNN) avec une seule sortie. Un CNN est un réseau de neurones multicouches formé de deux sections. La première section est une série de couches de convolutions, activation et echantillonage, semblable à la partie encodeur de l'auto-encodeur. La deuxième section est appelée « Fully Connected Layer », elle est formée d'une série de couches de vecteurs de neurones densément connectés.

Un des objectifs importants que nous avons fixés pour cette section de l'étude été d'utiliser l'auto-encodeur déjà entrainé pour améliorer la performance de notre classifieur. Pour achever cela, nous avons construit 3 modèles principaux.

1. Un modèle à base auto-encodeur : un modèle préentrainé par un auto-encodeur. C'est le modèle où nous avons concentré notre effort le plus afin d'améliorer sa performance.
2. Un modèle « from scratch » : un modèle sans préentrainement, qui sert comme une référence pour notre modèle à base auto-encodeur.
3. Un modèle à base de VGG16 : Un modèle où la partie encodeur est la partie encodeur de VGG16 (Simonyan et Zisserman, 2009) avec des poids entrainés sur ImageNet (Deng *et al.*, 2009). Ce modèle sert comme une référence de la meilleure performance que nous pouvons répliquer.

Facteurs contribuant à la performance

Durant notre étude nous avons expérimenté avec plusieurs techniques et plusieurs architectures afin d'améliorer la performance de notre classifieur. Nous avons isolé les facteurs principaux qui contribuent à la performance du modèle.

1. **La taille des images d'entrée :** Généralement, plus la taille des images d'entrée est grande plus la performance du classifieur augmente, car une image plus grande contient plus de détails. Ceci est clair dans le tableau [4.4] (les tests sont effectués sur un petit échantillon de 3000 images noires et blanches avec 20 itérations). Or, plus la taille des images augmente, plus de ressources sont nécessaires pour traiter les images, ceci est clair dans la colonne *Temps d'exécution* qui augmente exponentiellement avec la taille des images. Il est important de noter qu'on n'a pas pu performer des expérimentations sur des images de taille plus que 200x200, car les ressources de notre environnement ne sont pas suffisantes.

Taille de l'image	Temps (M)	Précision
16x16	0.63	62.5
32x32	0.84	68.1
64x64	1.68	70.2
128x128	5.9	75.6
200x200	14.5	72.2

TABLE 4.4 – Effet de la taille des images sur l'apprentissage

2. **Les couleurs des images d'entrée :** Nous avons trouvé avec consistance que les images colorées donnent de meilleurs résultats que les images noires et blanches, car, les images colorées sont plus riches en termes de temps de traitement et de mémoire. Ceci apparait évidement dans le tableau [4.5] où le classifieur avec des images colorées dépasse le classifieur noir et blanc quelque soit la taille des images, mais le temps de traitement du premier est plus grand que celui du second.

	Niveaux de gris		Colorés	
Taille de l'image	Temps (M)	Précision	Temps (M)	Précision
16x16	0.63	62.5	0.58	64.3
32x32	0.84	68.1	0.88	69.3
64x64	1.68	70.2	1.9	72.8
128x128	5.9	75.6	6.06	76.2
200x200	14.5	72.2	15.11	72.7

TABLE 4.5 – Effet des images colorées sur l'apprentissage

3. **La taille de l'échantillon d'entrainement :** Plus le nombre d'images d'entrainement augmente, plus la performance du classifieur va augmenter. Ceci est clair dans le tableau [4.6] où on remarque que la précision de notre modèle augmente avec la taille de l'échantillon d'apprentissage. Or, un grand nombre d'images nécessite beaucoup plus de ressources et de mémoire pour les traiter. Ceci est clair dans la colonne « Temps » du tableau [4.6].

Taille des données	Temps (H)	Précision
3,000	0.11	70.9
6,000	0.3	74.0
12,500	1.19	81.92
25,000	2.3	88.6

TABLE 4.6 – Effet de la taille de l'échantillon d'apprentissage

4. **La profondeur du réseau de neurones :** généralement, Plus le réseau est profond, plus la performance va augmenter (Srivastava *et al.*, 2015). Or, un réseau profond demande plus de ressources pour l'entrainement, car le nombre de paramètres va augmenter avec le nombre de couches. Dans notre étude nous avons adopté 28 couches pour la section de convolution et 2 couches pour la section densément connectée. Nous avons constaté qu'ajouter plus de couches dans la section densément connectée n'a pas affecté la performance.
5. **La fonction d'activation utilisée :** nous avons testé deux fonctions d'activations (Relu et Sigmoïde). Dans nos expérimentations, Relu a surpassée Sigmoïde dans la performance, comme le montre figure [4.7]. On a utilisé les mêmes paramètres pour les deux tests (Images colorées 128x128 avec 50 itérations). ReLu a donné une précision de 92.3%, alors qu'avec Sigmoïde cette valeur été de 89.7% . Or, pour la dernière couche, il est nécessaire qu'elle soit Sigmoïde, car la sortie est entre 0 et 1 (où 0 est Chat et 1 est Chien).

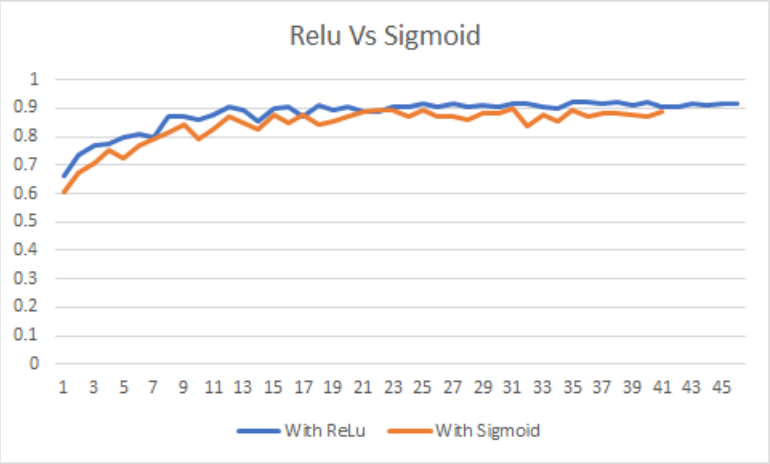

FIGURE 4.7 – Effets des fonctions ReLu et Sigmoïde

6. **Dropout :** Dropout est une opération qui débranche des neurones aléatoirement durant l'apprentissage. Elle aide à lutter le surapprentissage. Dans notre modèle nous avons utilisé dropout dans la section «Fully Connected» avec un pourcentage de 50%. D'après nos recherches, il est recommandé d'utiliser Dropout dans tout le réseau, même dans la section de convolution. Or d'après nos tests, ajouter Dropout à la section de convolution a nui à la performance.
7. **D'autres paramètres importants :**
   - L'optimiseur : Dans notre étude nous avons principalement utilisé RMSProp avec un taux d'apprentissage de 0.0001.
   - Le nombre d'itérations et la patience du early stopping : Généralement on a entrainé sur 20 ou 50 itérations avec une patience de 10 itérations.
   - La taille du batch de l'apprentissage : Dans notre étude nous avons utilisé un batch de 32 (à noter qu'un batch plus que 32 n'était pas possible sur notre environnement).

Architecture principale

L'architecture principale du classifieur est corrélée avec l'architecture de l'auto-encodeur, alors que la composition de la première section du classifieur est identique à de celle de l'auto-encodeur. Le modèle prend des images noirs et blancs de taille 128x128 avec 35 couches en total et 17,954,273 paramètres à entrainer. Les détails du modèle sont représentés dans la figure [4.8]
La première section est formée de 28 couches qui sont 4 successions de :

» Conv -> ReLu -> Conv -> ReLu -> Pooling.

La deuxième section est formée de 7 couches. La première couche est *Flatten* qui est un vecteur formé par l'étalement des matrices en sortie par la première section. Le vecteur passe ensuite par une deuxième couche densément connecté de taille 256. Les résultats de la deuxième couche sont passés par une fonction d'activation ReLu avec Normalization. La deuxième couche est connectée à une seule sortie. Le résultat final est passé par une fonction sigmoïde qui donne une valeur entre 0 et 1 : (Chat <0.5) et (Chien >0.5).

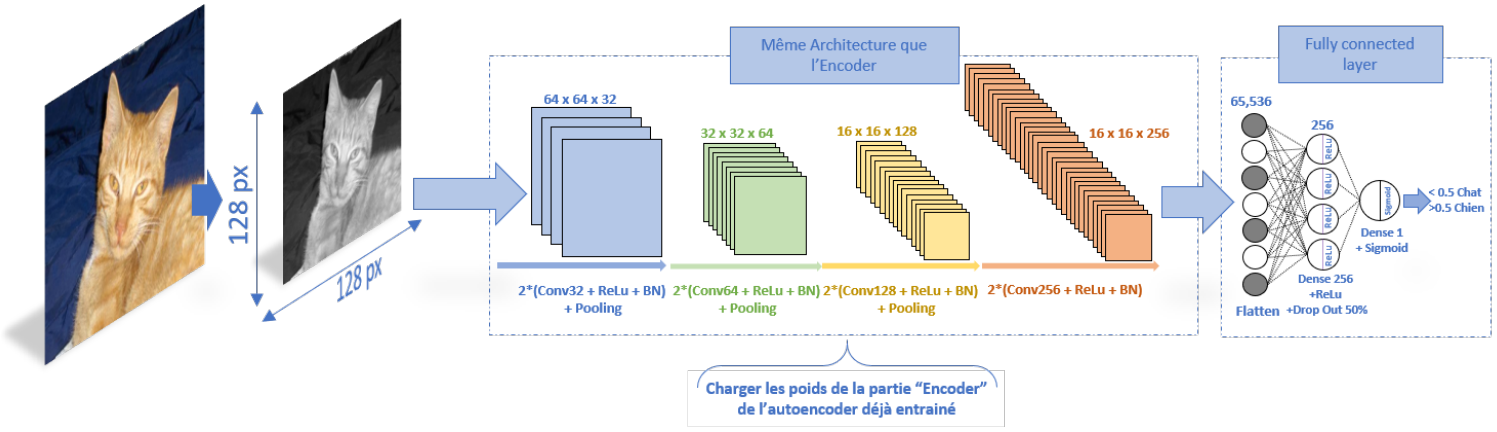

FIGURE 4.8 – Classifieur - Architecture principale

Modèle Basé auto-encodeur - Images en Noir et Blanc [128x128]

Introduction

Le modèle à base d'auto-encodeur est le modèle décrit dans la section Architecture principale. Le but de ce modèle est de pouvoir transférer les dimensions apprises par l'auto-encodeur au classifieur afin d'améliorer la performance de ce dernier.

Apprentissage

Pour transférer l'apprentissage de l'auto-encodeur au classifieur, après plusieurs tentatives nous avons adopter cette technique :

1. Les poids de la section encodeur de l'auto-encodeur sont chargés dans la même section du classifieur et puis ces couches sont figées (elles ne vont pas être entrainées).
2. Le modèle est entrainé sur 20,000 images et validé sur 5,000 images (les détails sont présentés dans le tableau [4.7]). Seule la section «Fully Connected» va être entrainée comme la première section est figée.
3. Les poids du modèle sont sauvegardés après l'entrainement.
4. Fine Tuning : Les poids de l'étape précédente sont rechargés dans le modèle et toutes les couches sont défigées. Le modèle est entièrement entrainé avec un taux d'apprentissage très petit pour ne pas abimer les poids déjà entrainés.

Le meilleur résultat que nous avons obtenu est d'une précision de 87.1% (Figure [4.9]) avec une perte de 0.408 (Figure [4.9]). Le modèle était entrainé sur 50 itérations avec une patience de 10. L'entrainement a pris 2.3 heures. Les paramètres aussi que les résultats sont détaillés dans le tableau[4.8].

	Apprentissage	**Validation**	**Total**
Chats	10,000	2,500	12,500
Chiens	10,000	2,500	12,500
			25,000

TABLE 4.7 – Modèle Basé Auto-encodeur - Données

Paramétres							
Couleur	**Largeur**	**Taille**	**Itérations**	**Patience**	**Activation**	**Params**	**couches**
Gris	128	128	50	10	ReLu	16,778,241	35
Resultats							
Précision		**Perte**		**Temps (H)**		**E. Stopping**	
87.1		**0.408**		2.3		NA	

TABLE 4.8 – Modèle Basé Auto-encodeur - Paramètres et Résultats

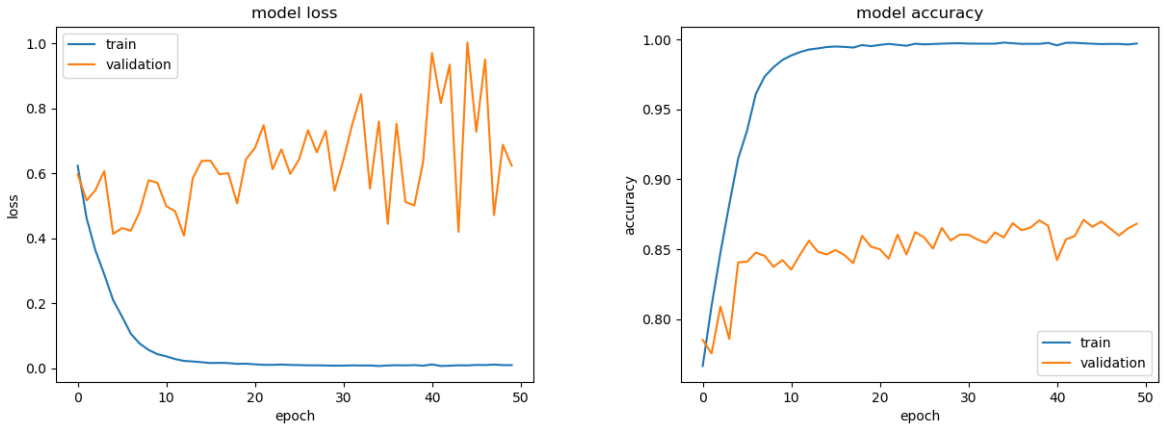

FIGURE 4.9 – Modèle Base auto-encodeur - Courbe de Loss / Courbe de Accuracy

Résultats sur les images de tests

Le modèle est testé sur 5000 images divisées en 2500 images de Chiens et 2500 images de chats. Le modèle avait une accuracy de 87.6%. De l'ensemble des 5000 images de test, le modèle a classé 4380 images correctement et a mal classé 619 images. La matrice de confusion [4.10] montre le nombre d'images où le modèle a confondu les chats avec des chiens et vice-versa. La courbe ROC [4.10] représente la précision de notre modèle à séparer les chiens des chats avec une valeur AUC de 0.876.

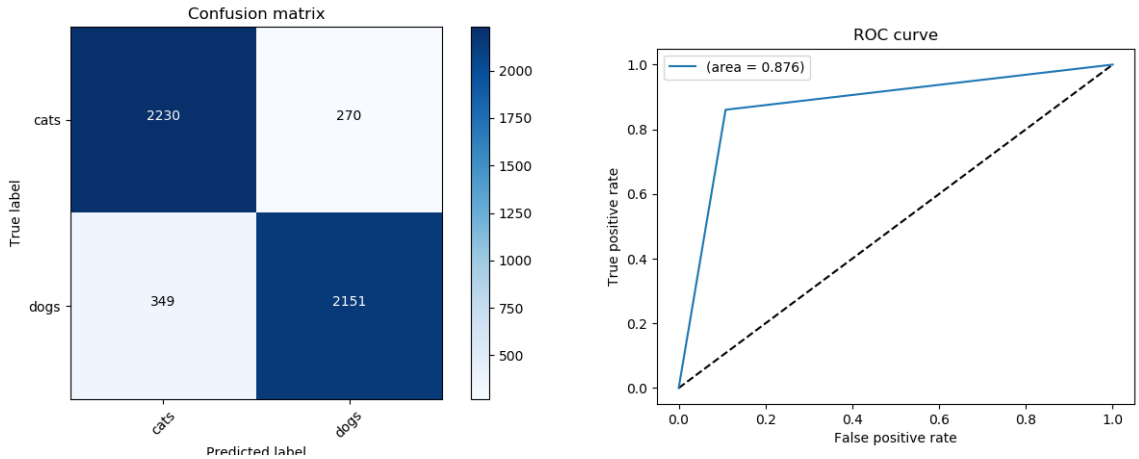

FIGURE 4.10 – Modèle Basé Auto-encodeur - Matrice de Confusion / Courbe ROC

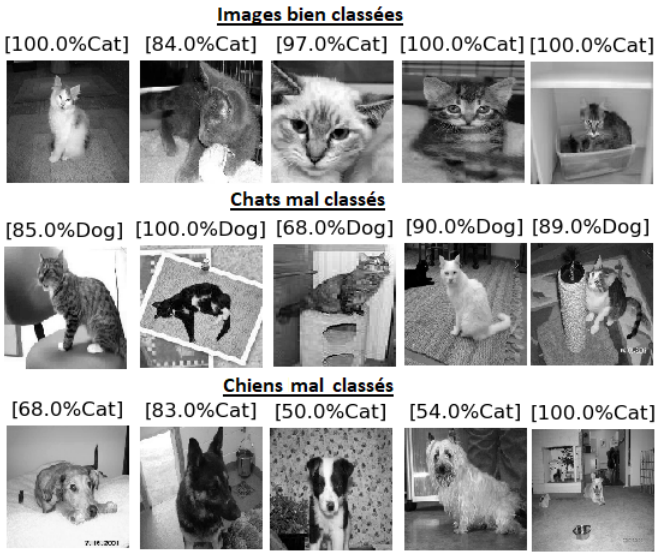

FIGURE 4.11 – Modèle Basé Auto-encodeur - Échantillons de Test

Conclusion sur l'efficacité de l'auto-encodeur

Pour bien tester l'efficacité de l'auto-encodeur à améliorer l'apprentissage du classifieur, nous avons performé une série de 12 expérimentations. Les résultats de ces expérimentations sont détaillés dans le tableau [4.10].

Nous avons utilisé le modèle principal (Images noires et blanches de taille [128x128]) avec les mêmes paramètres et architectures pour toutes les expérimentations. Les modèles étaient entrainés sur 50 epochs avec une patience de 10. Nous avons comparé 3 variations du même modèle :

1. Pour la première variation, nous avons utilisé les poids de l'auto-encodeur pour le classifieur sans Tuning.
2. Pour la deuxième variation, nous avons utilisé les poids de l'auto-encodeur avec une phase de Tuning.
3. Pour la troisième variation, nous avons entrainé le classifieur « From Scratch ».

Les expérimentations étaient divisées en 4 phases. Pour la première phase, nous avons entrainé les 3 variations du modèle sur 2000 images avec 1000 images de validation. Puis, nous avons augmenté graduellement le nombre des images d'apprentissage pour chaque phase. Le précision de la validation a été sauvegardé pour chaque modèle avec le temps d'exécution.

D'après le tableau [4.10], nous remarquons que les modèles avec le auto-encodeur sans ajustements (1,4,7,10) sont les pires. Les modèles avec auto-encodeur avec ajustements sont les meilleurs pour les 3 premières phases. Alors que pour la dernière phase (20,000 images), le modèle « From Scratch » est le meilleur. Nous concluons que plus le nombre de données étiquetées augmente, plus le pouvoir de l'auto-encodeur à améliorer la performance du classifieur diminue. Ceci est très clair dans le tableau [4.9], où on remarque une amélioration de 6.3% avec les 2,000 images, mais cette valeur diminue graduellement jusqu'à la quatrième phase où le modèle « From scratch » dépasse le modèle pré-entrainé par 1.5%.

	Sans AE	Avec AE ajusté	Améliorations
Phase 1 (2,000)	0.709	0.772	**+6.30%**
Phase 2 (5,000)	0.74	0.769	**+2.90%**
Phase 3 (10,000)	0.8192	0.824	**+0.48%**
Phase 4 (20,000)	0.886	0.871	**-1.50%**

TABLE 4.9 – Expérimentations sur l'effet de l'Auto-encodeur - Résumé

	Paramètres					
	Couleur	**Taille**	**Itérations**	**Patience**	**Paramètres**	**Couches**
	Niveaux de gris	128x128	50	10	16,778,241	35
	Phase 1 (2,000)					
	Modèle	**Prec. Val.**	**Avec AE**	**Données**	**Early Stop**	**Temps(h)**
1	AvecAE	**0.7**	Oui	2000/1000	31	0.08
2	Avec AE Ajusté	**0.772**	Oui	2000/1000	14	0.07
3	Sans AE	**0.709**	Non	2000/1000	21	0.11
	Phase 2 (5,000)					
4	Avec AE	**0.746**	Oui	5000/1000	26	0.19
5	Avec AE Ajusté	**0.769**	Oui	5000/1000	15	0.28
6	Sans AE	**0.74**	Non	5000/1000	23	0.3
	Phase 3 (10,000)					
7	Avec AE	**0.7632**	Oui	10000/2500	23	0.29
8	Avec AE Ajusté	**0.824**	Oui	10000/2500	29	0.76
9	Sans AE	**0.8192**	Non	10000/2500	Non	1.19
	Phase 4 (20,000)					
10	Avec AE	**0.7762**	Oui	20000/5000	23	0.52
11	Avec AE Ajusté	**0.871**	Oui	20000/5000	47	2.3
12	Sans AE	**0.886**	Non	20000/5000	Non	2.3

TABLE 4.10 – Expérimentations sur l'effet de l'auto-encodeur - Détails

Modèle from scratch - Images en Couleurs [150x150]

Introduction

Le modèle coloré « from scratch » est entrainé avec la même architecture que le modèle Noir et Blanc mais sans les couches de normalisation du batch. Nous avons remarqué dans nos tests que le modèle performait un peu mieux sans la normalisation du batch. À noter que ceci est contradictoire avec les recommandations standards. Le modèle contient 28 couches avec 7,726,369 paramètres à entrainer.

Apprentissage

Le modèle est entrainé sur 20,000 images et validé sur 5,000, les détails des données sont explicités dans le tableau [4.11]. L'entrainement s'est fait sur 50 itérations avec une patience de 10. Le modèle a arrêté sur la 49e itération avec une précision de validation de 93.38% (figure [4.12]) et une perte dur la validation de 0.198 (figure [4.12]). Les paramètres utilisés pour l'apprentissage ainsi que celles des résultats sont détaillés dans le tableau [4.12].

	Apprentissage	**Validation**	**Total**
Chats	10,000	2,500	12,500
Chiens	10,000	2,500	12,500
			25,000

TABLE 4.11 – Modèle From Scratch - Données

Paramètres							
Couleur	**Largeur**	**Taille**	**Itérations**	**Patience**	**Activation**	**Param.**	**Couches**
RGB	150	150	50	10	ReLu	7,726,369	28
Resultats							
Précision		**Perte**		**Temps (H)**		**Early Stopping**	
93.38		**0.198**		2.5		49e itération Epoch	

TABLE 4.12 – Modèle From Scratch - Paramètres et Résultats

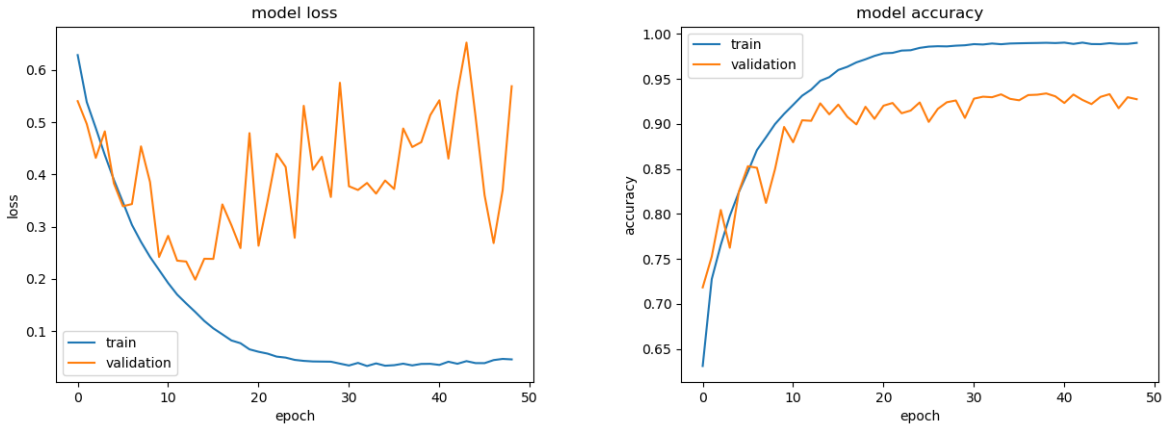

FIGURE 4.12 – Modèle From Scratch - Courbe de Perte / Courbe de Précision

Résultats sur les images de test

Le modèle est testé sur 5000 images divisées en 2500 images de chiens et 2500 images de chats. Le modèle avait une accuracy de 92.88%. De l'ensemble des 5000 images de test, le modèle a classé 4643 images correctement et a mal classé 356 images. La matrice de confusion [4.13] montre le nombre d'images où le modèle a confondu les chats avec des chiens et vice-versa. La courbe ROC [4.13] représente la précision de notre modèle à séparer les chiens des chats avec une valeur AUC de 0.929.

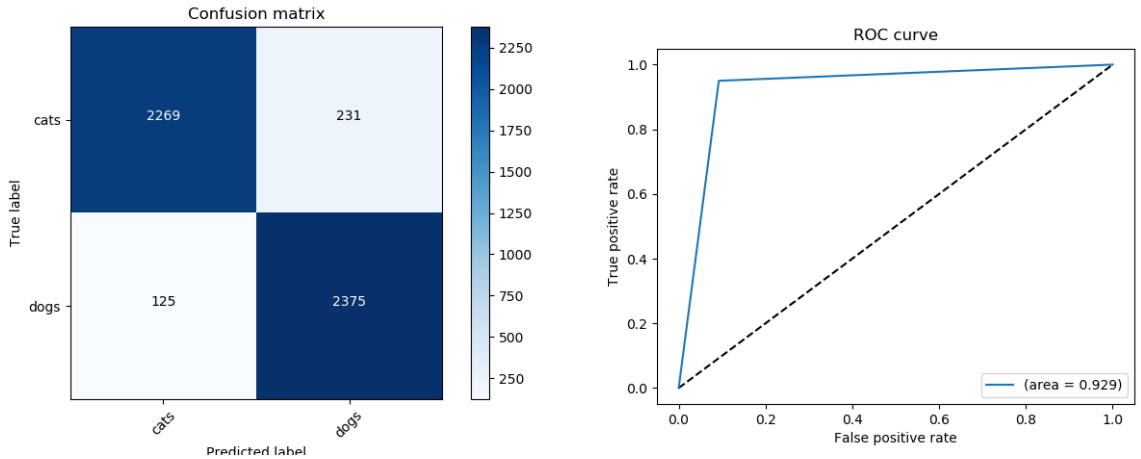

FIGURE 4.13 – Modèle From Scratch - Matrice de Confusion / Courbe ROC

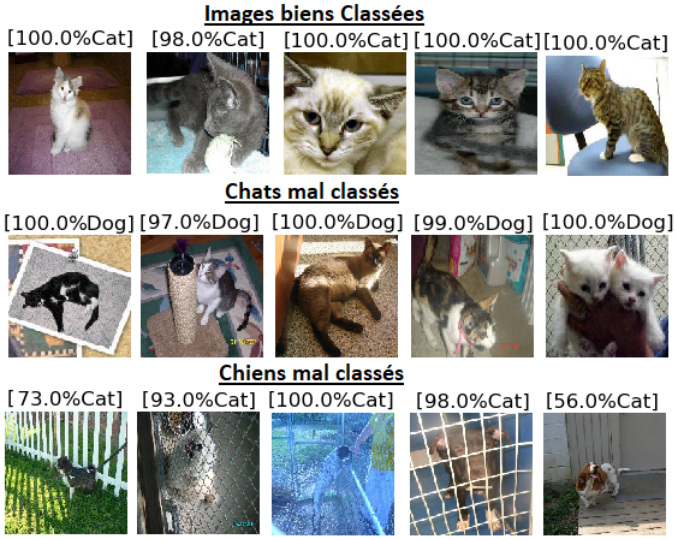

FIGURE 4.14 – Modèle From Scratch - Échantillons de Test

Interprétation des résultats

Ce modèle a montré une amélioration de 7% par rapport au modèle noir et blanc. Ceci est dû aux facteurs suivants :

1. Images colorées Vs Images noires et blanches.
2. La taille des images d'entrée est plus grande ($150 > 128$).
3. La taille de l'échantillon d'apprentissage est très grande. Comme c'était mentionné dans la section précédente, plus l'échantillon d'images étiquetées augmente plus la performance du modèle « from scratch » dépasse le modèle pré-entrainé par l'auto-encodeur.

Le modèle performe très bien sur les images tests et ne trompe que dans les images où les chats et les chiens sont derrière des barrières où quand les couleurs ne sont pas claires, comme le montre la figure [4.14].

Modèle Basé sur VGG16 - Images en Couleurs [200x200]

Introduction

Afin de comparer nos résultats à une référence de Benchmark, nous avons développé un modèle basé sur VGG16 (Simonyan et Zisserman, 2009) et entrainé sur ImageNet (Deng *et al.*, 2009). VGG16 est un modèle très profond développé par (Simonyan et al.). Le modèle a 41 couches avec 138 millions paramètres. ImageNet est une base de données d'un million d'images distribuées suivant une hiérarchie de plusieurs classes. VGG16 a été entrainé sur ImageNet pour la compétition « ImageNet Challenge 2014 » et s'est classé seconde derrière GoogLeNet (Szegedy *et al.*, 2014).

Apprentissage

Pour entrainer le modèle, nous avons adopté la technique suivante :

1. Entrainement initial :
   (a) Nous avons pris la section encodeur de VGG16
   (b) Les Images colorées de taille 200x200 sont prétraitées en soustrayant la moyenne $[103.939, 116.779, 123.68]$ des Images (avec cette technique on a eu une amélioration de 2%).
   (c) Les images d'apprentissage et de validation sont passées par l'encodeur de VGG16 et le résultat est sauvegardé comme « Numpy Arrays ».
   (d) Nous avons construit un modèle « Fully Connected » de 3 Couches (Flatten $>$ 256 $>$ 1) avec un Dropout de 50
   (e) Le modèle est entrainé sur les « Numpy Arrays » avec un optimiseur RMSProp et un taux d'apprentissage de 0.0001.
   (f) Les poids finals de la phase 1 du modèle entrainé sont sauvegardés.
2. Ajustements :
   (a) L'encodeur du VGG est rattaché à la section « Fully Connected » et la section encodeur est figée.
   (b) Les poids déjà sauvegardés dans la section précédente sont chargés dans la section « Fully Connected ».
   (c) Le modèle est entrainé avec les images initiales.
   (d) Les poids du modèle final sont sauvegardés.

Le modèle est entrainé sur 20,000 images et validé sur 5,000, les détails des données sont détaillés dans le tableau [4.13]. L'entrainement s'est fait sur 50 itérations avec une patience de 10. Le modèle s'arrête à la 49e itération avec une précision de validation de 98.2% (figure [4.16]) et une Validation Loss de 0.1256(figure [4.16]). Les paramètres utilisés pour l'apprentissage ainsi que les résultats sont détaillés dans le tableau [4.14].

	Apprentissage	**Validation**	**Total**
Chats	10,000	2,500	12,500
Chiens	10,000	2,500	12,500
			25,000

TABLE 4.13 – Modèle Base de VGG16 - Données

Paramètres							
Couleur	**Largeur**	**Taille**	**Itérations**	**Patience**	**Activation**	**Param.**	**Couches**
RGB	200	200	50	10	ReLu	4,719,105	43
Résultats							
Précision		**Perte**		**Temps (H)**		**Early Stopping**	
98.2		**0.1256**		NA		29	

TABLE 4.14 – Modèle Basé sur VGG16 - Paramétrés et Résultats

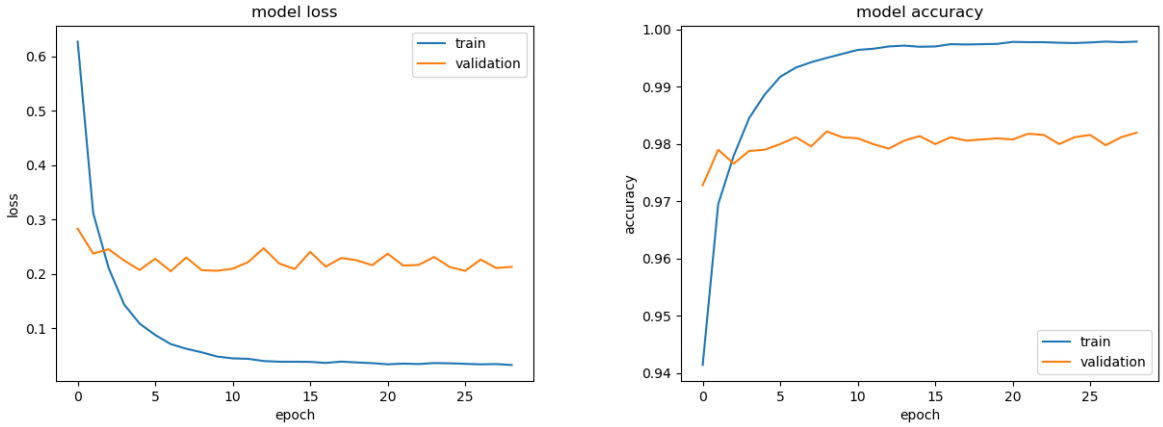

FIGURE 4.15 – Modèle Basé VGG16 Phase 1 - Courbe de perte / Précision

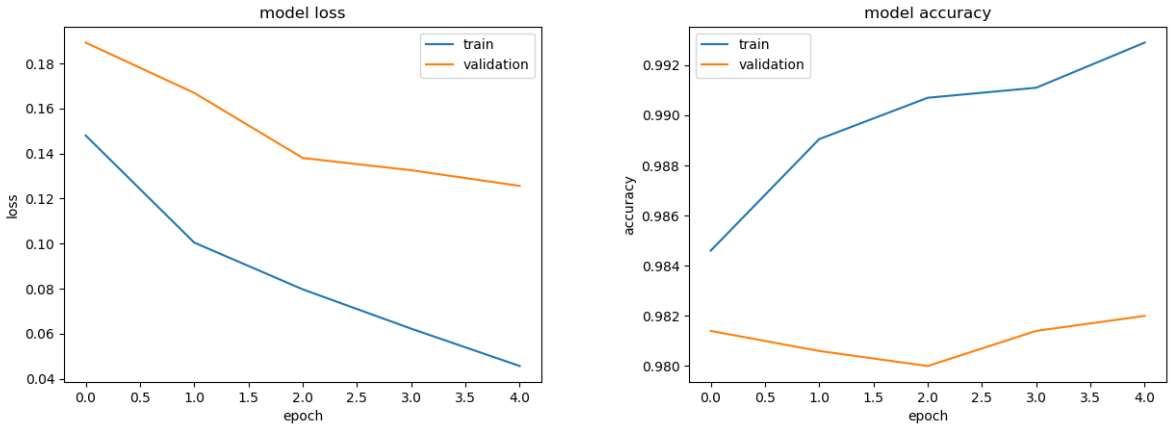

FIGURE 4.16 – Modèle Basé sur VGG16 Ajusté - Courbe de perte/ Précision

Résultats sur les images de test

Le modèle est testé sur 5000 images divisées en 2500 images de chiens et 2500 images de chats. Le modèle avait une précision de 97.52%. De l'ensemble des 5000 images de test, le modèle a classé 4880 images correctement et a mal classé 119 images. La matrice de confusion [4.17] montre le nombre d'images où le modèle a confondu les chats avec des chiens et vice-versa. La courbe ROC [4.17] représente la précision de notre modèle à séparer les chiens des chats avec une valeur AUC de 0.976.

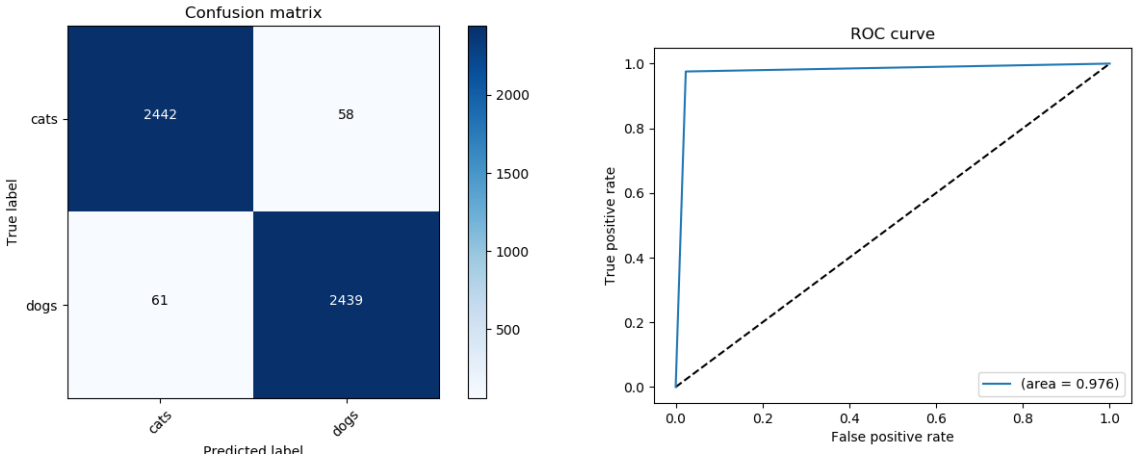

FIGURE 4.17 – Modèle Base VGG16 - Matrice de Confusion / Courbe ROC

FIGURE 4.18 – Modèle Base VGG16 - Échantillons de Test

4.3.4 Interprétation des résultats

Le modèle à base VGG16 a donné les meilleurs résultats. Il a même détecté les fautes de l'étiquetage manuel (la 8ème image de la figure [4.18]). Ces bons résultats sont dus aux facteurs suivant :

1. Les images sont d'une grande taille de 200x200 (comme la partie encodeur est déjà entrainée, nous économisons beaucoup de mémoire et on n'a pu augmenter la taille des images).
2. VGG16 est très profond.
3. VGG16 est entrainé sur ImageNet qui contient des millions d'images (incluant des images des chats et des chiens).

4.4 Coyotes Vs Chiens

4.4.1 Préparation des données

Le problème principal avec la partie de la classification des coyotes et des chiens été l'absence des images de coyotes. Pour faire face a ce problème, nous avons décidé d'appliquer la technique de « Scrapping ». Cette technique consiste à fouiller les sites de recherche d'images comme *Google* et *Bing* afin de télécharger les résultats. Comme le nombre d'images nécessaire pour l'apprentissage est très grand, nous avons automatisé cette partie.

Notre technique consistait à utiliser *Internet Explorer* en mode de développement pour fouiller *Google* et *Bing* Images, puis sauvegarder le log de la recherche comme un fichier à extension « .har ». Les résultats de recherche sont sauvegardés sous format *Json* dans ce fichier. Puis, nous avons développé une petite application avec Python pour lire les fichiers « .har » et détecter toutes les URL des images dans ces fichiers en utilisant *Regex*. Une fois les URL sont extraites, notre programme téléchargeait les images référencées par les URL séquentiellement. Quand toutes les images étaient téléchargées, nous les avons nettoyés manuellement. Nous avons abouti avec 5,532 images de coyotes.

Pour les images des chiens, nous avons pris 5,532 images de la base de données de *Kaggle*. Mais les images étaient choisies à partir de l'échantillon de test et non pas de celui de l'apprentissage ou de la validation. On a pris cette approche, car nous allons utiliser le transfert d'apprentissage et il est nécessaire que notre modèle n'ait pas déjà vu ces images, sinon on risque le surapprentissage.

4.4.2 Entraînement avec Transfert d'Apprentissage

Pour entrainer notre classifieur à distinguer entre les chiens et les coyotes, nous utilisons la technique de Transfer Learning. Plusieurs méthodes existent pour effectuer ce Transfert d'apprentissage ; pour notre étude, nous utilisons la technique de transfert des poids. À partir les modèles déjà entrainer sur les chats et les chiens, nous choisissons le modèle « Images colorées – From Scratch 150x150 » noté modèle X.

Nous construisons un modèle Y avec la même architecture que le modèle X. Les poids des 20 premières couches (couches de convolutions) du modèle X sont chargés dans le modèle Y et ces couches sont figées (leurs poids ne vont pas changer durant l'apprentissage). Durant l'apprentissage, seulement la partie « Fully Connected » va être entrainée.

L'entrainement s'est effectué sur 11,064 images (5,532 Coyotes et 5,532 Chiens). Nous divisons les images de chaque classe comme suite (détails tableau [4.15]) : 4,750 images de training, 532 images de validation et 250 images de test. Le classifieur est entrainé avec des images colorées de taille 150x150. Il réalise une validation accuracy de 93.23%[4.19] avec une validation loss de 0.199[4.19] dans 27 itérations. Les paramètres utilisés aussi que les résultats sont détaillés dans le tableau[4.16].

	Apprentissage	Validation	Tests	Total
Coyotes	4,750	532	250	5,532
Chiens	4,750	532	250	5,532
				11,064

TABLE 4.15 – Coyotes vs Chiens - Détails des données

Paramètres							
Couleur	**Largeur**	**Taille**	**Itérations**	**Patience**	**Activation**	**Param.**	**Couches**
RGB	150	150	50	10	ReLu	7,727,393	28
Résultats							
Précision		**Perte**		**Temps (H)**		**Early Stopping**	
93.23		**0.199**		0.16		27e itération	

TABLE 4.16 – Coyotes vs Chiens - Paramètres et Résultats

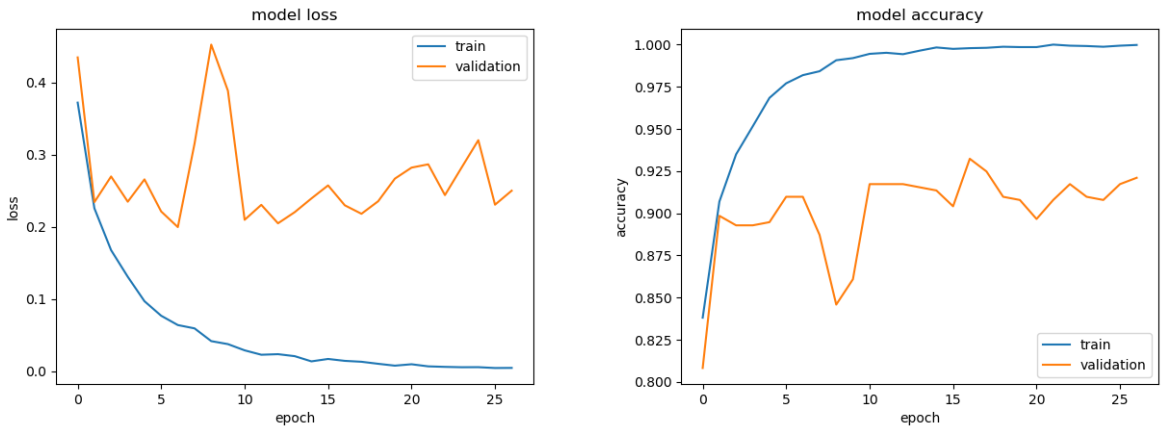

FIGURE 4.19 – Coyotes vs Chiens - Courbe de perte / Courbe de précision

4.4.3 Résultats sur les images de test

Le modèle est testé sur 500 images divisées en 250 images de Chiens et 250 images de Coyotes. Le modèle avait une précision de 89%. De l'ensemble des 500 images de test, le modèle a classé 444 images correctement et a mal classé 55 images. La matrice de confusion [4.20] montre le nombre d'images où le modèle a confondu le coyote avec un chien et vice-versa. La courbe ROC [4.20] représente la précision de notre modèle à séparer les chiens des coyotes avec une valeur AUC de 0.89.

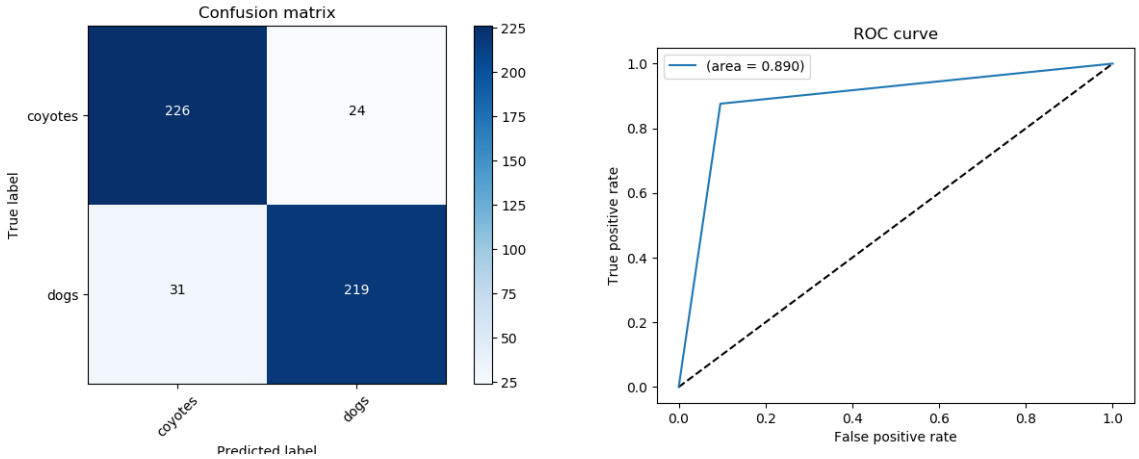

FIGURE 4.20 – Coyotes vs Chiens - Matrice de Confusion / Courbe ROC

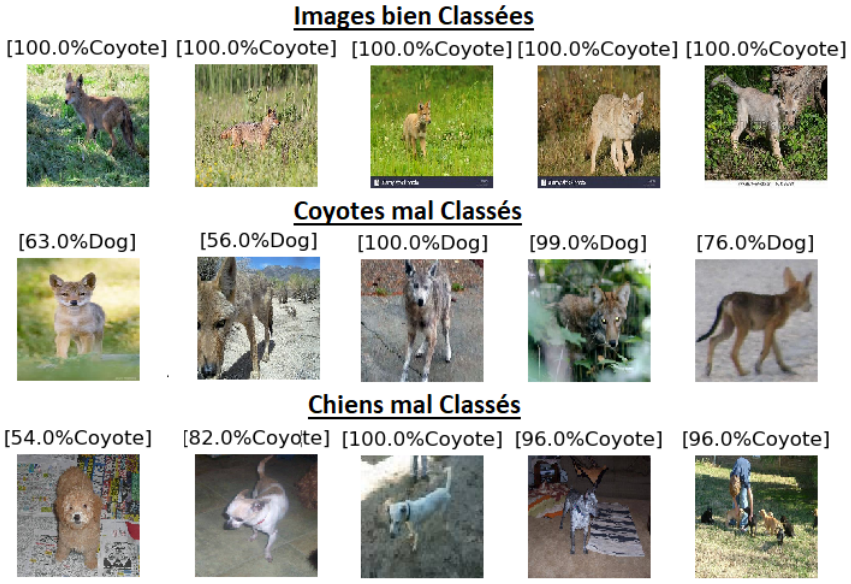

FIGURE 4.21 – Coyotes vs Chiens - Échantillons de Test

4.4.4 Interprétation des résultats

Le bon résultat de notre modèle, avec relativement un petit échantillon de données, est dû au transfert d'apprentissage. Les poids déjà entrainés dans la phase des Chats et des Chiens ont bien aidé notre classifieur à identifier les chiens. Cependant, nous suspectons que les bons résultats ne sont pas dus au pouvoir du classifieur à distinguer entre les Chiens et les Coyotes plutôt, que sur sa capacité à distinguer entre les chiens et d'autres sujets. En d'autres termes, le modèle n'a pas vraiment capturé les traits de ce qui constitue un coyote. Ceci est clair dans la figure [4.21] où on constate que les photos où le classifieur a confondu les chiens à des coyotes sont vraiment des photos de petits chiens de maison qui se ressemble à des chats.

4.5 Conclusion

À la fin de ce chapitre, nous avons fait le tour de l'implémentation réalisé dans le cadre de notre projet de recherche.

Conclusion générale

Dans le présent travail de session pour le cours INF-7370, nous avons tout d'abord cherché une problématique principale parmi ceux cités dans le document de proposition de recherche. Après plusieurs discussions avec le directeur de recherche, nous avons pu fixer une problématique primaire qui sera rattachée au sujet de la distinction entre chats et chiens en utilisant des modèles d'apprentissage profond, mais qui toutefois serait extensible vers un problème de distinction entre chiens et coyotes qui demeure une problématique très peu évoquée dans les travaux de l'état de l'art spécifiquement à cause de l'absence d'un nombre d'images important et étiquetées permettant de faire face au problème.

En premier lieu, nous avons commencé par faire des recherches à ce sujet, ces dernières nous ont emmenés vers une solution en utilisant les réseaux de neurones convolutionnels afin de distinguer les chats et chiens. Après cela, nous avons lu et recherché des articles qui concernaient l'extraction des caractéristiques à partir de l'utilisation des auto-encodeurs et l'utiliser directement dans un réseau de neurones.

En dernier, nous avons lu et exploré quelques solutions quant à l'utilisation de l'apprentissage par transfert pour effectuer la distinction entre chiens et coyotes. Par la suite vient l'étape de l'implémentation, nous avons utilisé principalement les auto-encodeurs avec plusieurs modélisations avec des images en couleurs et en noir et blanc. Les caractéristiques des chats et chiens étant définies, nous avons par la suite effectué une classification avec un réseau de neurones convolutionnel. Tous les résultats récoltés lors de la section précédente ont été comparés à un modèle référence VGG-16.

Enfin, nous avons implémenté le modèle source avec la meilleure performance en l'occurrence le classifieur « from scratch » pour distinguer les chiens et les coyotes puisque le processus décisionnel ne dépend que du modèle utilisé dans la première partie et que la décision dans ce cas était plutôt soit chien ou non chien. L'objectif principal de ce travail de recherche était d'exploiter des travaux de l'état de l'art et explorer des solutions nouvelles ; l'objectif étant atteint, cela nous a permis d'acquérir une base théorique et pratique suffisante pour explorer de nouveaux travaux dans ce domaine.

Toutefois, il est très important de signaler que la solution proposée reste largement perfectible. L'une des perspectives serait de construire un ensemble de données contenant des images choisies par des experts afin de distinguer les coyotes des autres espèces animales ; mais qui reste applicable uniquement dans ce domaine.

Bibliographie

Bang Liu, Y. L. e. K. Z. (2014). *Image Classification for Dogs and Cats*. Rapport technique, University of alberta.

Collab, G. (2019). Google collab. Récupéré de `https://colab.research.google.com/notebooks/welcome.ipynb`

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. et Fei-Fei, L. (2009). ImageNet : A Large-Scale Hierarchical Image Database. Dans *CVPR09*.

Elson, J. E., Douceur, J. R., Howell, J. et Saul, J. (2007). Asirra : a captcha that exploits interest-aligned manual image categorization. Dans *ACM Conference on Computer and Communications Security*.

Ioffe, S. et Szegedy, C. (2015). batch normalization : accelerating deep network training by reducing internal covariate shift.

Kaggle (2013). dogs vs. cats competition. Récupéré de `https://www.kaggle.com/c/dogs-vs-cats`

Keras (2019). Keras. Récupéré de `https://keras.io`

Pan, S. J. et Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. `http://dx.doi.org/10.1109/TKDE.2009.191`

Research, M. (2012). Assira. Récupéré de `http://research.microsoft.com/en-us/um/redmond/projects/asirra`

Simonyan, K. et Zisserman, A. (2009). very deep convolutional networks for large-scale image recognition.

Srivastava, R. K., Greff, K. et Schmidhuber, J. (2015). Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, et R. Garnett (dir.), *Advances in Neural Information Processing Systems 28* 2377–2385. Curran Associates, Inc.

Stanford (2018). Autoencoders. Récupéré de `http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/`

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. et Rabinovich, A. (2014). Going deeper with convolutions.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. et Liu, C. (2018). A survey on deep transfer learning. *CoRR*, *abs/1808.01974*. Récupéré de `http://arxiv.org/abs/1808.01974`

Yosinski, J., Clune, J., Bengio, Y. et Lipson, H. (2014). How transferable are features in deep neural networks ? *CoRR*, *abs/1411.1792*. Récupéré de `http://arxiv.org/abs/1411.1792`