UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR
M'BEGNAN NAGNAN ARTHUR

DÉVELOPPEMENT D'OUTILS WEB DE DETECTION D'ANNOTATIONS MANUSCRITES DANS LES IMPRIMÉS ANCIENS

AVRIL 2021

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Fadel Toure, directeur de recherche
Département de mathématiques et d'informatique, UQTR

M. Alain Goupil, membre du jury
Département de mathématiques et d'informatique, UQTR

M. Usef Faghihi, membre du jury
Département de mathématiques et d'informatique, UQTR

REMERCIEMENTS

Je tiens avant tout à remercier mon Dieu dans le nom de Jésus par lequel je suis arrivé jusqu'ici; qui m'a soutenu, relevé de tous mes échecs et conduit depuis ma présence dans ce pays et dans cette ville de Trois-Rivières. Je remercie ma tendre et merveilleuse épouse et mes parents qui m'ont apporté un soutien moral, affectif et financier hors pair.

Ces deux années ont été pour moi très enrichissante en ce sens qu'elles m'ont permis de faire la rencontre de personnes extraordinaires mais aussi d'acquérir de nombreuses compétences.

Mon directeur de recherche qui est Monsieur Fadel Toure PhD. m'a été d'un soutien infaillible car il m'a accompagné, encouragé, motivé et donné de son temps, de ses conseils durant toute la durée de mes études. Il m'a permis d'apprendre énormément dans le domaine de l'apprentissage profond et de la programmation. Je tiens aussi à remercier Monsieur Marc André Bernier qui est mon codirecteur de recherche; un monsieur très aimable et courtois qui a mis à ma disposition des ressources humaines dont madame Valérie Plourde qui m'a été d'une aide précieuse dans l'aboutissement de mes recherches mais aussi des ressources financières qui ont rendues mes recherches possibles.

Je remercie Monsieur Mourad Badri; un professeur que j'affectionne beaucoup en raison de son charisme, sa rigueur, son dévouement à nous enseigner et à nous communiquer de nouvelles connaissances. J'ai particulièrement aimé l'un de ses cours de la session d'hiver qui a approfondi mes connaissances en Software Engineering et qui m'a permis non seulement de me dépasser mais aussi m'a initié à la recherche.

Je tiens en dernier lieu à remercier tous les employés de l'UQTR, tous mes amis et toutes ces personnes qui de près ou de loin m'ont apporté leur aide et la directrice de mon département qui est madame Linda Badri qui, dès mon arrivé dans cette université, a été à mon écoute, a su me guider, m'orienter vers des personnes et dissiper certaines de mes craintes.

RÉSUMÉ

Un peu partout dans le monde, la recherche des deux dernières décennies dans le domaine des humanités numériques a été caractérisée par de nombreux travaux portant sur l'histoire du livre et de l'imprimé. Au Québec plus particulièrement, ces travaux ont cherché, dans un premier temps, à produire des analyses quantitatives, c'est-à-dire destinées à recenser les livres présents sur le territoire québécois; puis, dans un second temps, à créer des outils permettant d'effectuer des analyses qualitatives susceptibles de favoriser une meilleure exploitation de ces fonds. L'étude des *marginalia*, c'est-à-dire les annotations manuscrites, s'inscrit dans ce second moment. Les *marginalia* renseignent en effet sur les pratiques de lecture et renferment de nombreux indices sur l'histoire de la circulation du livre entre différents individus ou diverses institutions.

Le travail manuel qu'exige la numérisation, la classification et l'identification des annotations manuscrites et des marques de possession est couteux en temps et ressources. C'est dans ce contexte que nous avons apporté notre contribution en termes d'outils informatique permettant d'assister les acteurs. Ainsi, nous avons développé une application permettant entre autres de déterminer pour chacune des pages d'un livre la présence ou non d'annotations manuscrites, de manière à effectuer une classification de ces pages en deux catégories : *Annotée* et *Non-annotée*. Pour y arriver nous avons effectué plusieurs expérimentations sur le VGG16 pré-entrainé sur ImageNet à l'aide du Transfer Learning. Ce modèle est un réseau de neurones convolutif qui a fait ses preuves dans le domaine de la classification de documents, mais aussi dans le domaine des annotations manuscrites. Pour chacune de ces expérimentations, nous avons utilisé des techniques telles que l'augmentation de données, le *« Dropout »* et nous avons testé différentes tailles de lot. Nous avons obtenu un modèle qui a un taux d'exactitude de 93.75% sur nos données de validation et de 93.33% sur nos données. Le modèle présentant ces résultats a ensuite été déployé grâce à un package Flask et été intégré à notre application pour la classification automatique des pages des livres. Notre application intègre également la bibliothèque JavaScript Tesseract.js, que nous avons utilisée pour la transcription textuelle des images des pages numérisées des livres.

ABSTRACT

Around the world, the research of the last two decades in the field of digital humanities has been characterized by much work on the history of books and the print. In Quebec more particularly, this work sought, initially, to produce quantitative analyzes, which, aimed at identifying the books present in Quebec; then, secondly, to create tools allowing to carry out qualitative analyzes likely to favor a better exploitation of these funds. The study of *marginalia*, also called handwritten annotations, fall within this second period. *Marginalia* provides in fact informations on reading practices and contains many insights about the history of the circulation of books between different individuals or various institutions.

Scanning, classifying and identifying books with handwritten annotations pages is a time and resource consuming task. In such context, we have developed a tool which can determine for each scanned book pages, the presence or absence of handwritten annotations, by classifying it into 2 categories. To achieve this goal, we performed several experiments using the VGG16 model pre-trained on ImageNet dataset using Transfer Learning. This model is a convolutional neural network which has shown great performances in the field of document classification but also in the field of handwritten annotations. For each of these experiments; we used techniques such as data augmentation, Dropout and we tested different batch sizes. We were able to get a model with good generalization performances; we got an accuracy of 93.75% on validation data and 93.33% on test data by using batches of different sizes on our training and validation data. The obtained model was then deployed using a Flask package and integrated into our application for the automatic classification of the book pages. Our application also integrates Tesseract.js which is used for extracting texts from the scanned book pages.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1. INTRODUCTION

1.1 - Objectifs et problématiques

De nombreux travaux sur l'histoire du livre se sont développés un peu partout dans le monde depuis les ouvrages pionniers d'Henri-Jean Martin et Roger Chartier sur l'*Histoire de l'édition française*[1]. C'est ainsi que cette histoire se donnait comme tâche de faire du livre un objet scientifique et historique autonome, permettant aux professionnels du livre, aux bibliothécaires et aux éditeurs de mieux connaitre le produit dont ils assurent la fabrication, la diffusion ou la conservation. On retrouve également au Québec et au Canada des travaux qui ont été réalisés dans le même esprit, avec notamment le grand chantier qu'ont animé Fiona Black et Yvan Lamonde avec la série d'ouvrages consacrés à l'*Histoire du livre et de l'imprimé au Canada*[2]. C'est dans ce contexte que les travaux sur l'étude du livre au Québec examinent notamment ce qu'apprend, à la recherche actuelle en histoire culturelle, la place et le rôle que jouent les collections anciennes dans les institutions d'enseignement québécoises. Ces imprimés anciens et européens permettent, en effet, de mieux comprendre la dynamique des transferts de savoirs entre l'Ancien et le Nouveau Monde, de manière à retracer la formation, au Québec, d'une culture savante à la fois littéraire et philosophique. Ils renseignent aussi sur l'histoire de l'éducation, en indiquant quelles étaient les sources livresques de l'enseignement effectivement donné en classe. Par exemple, la bibliothèque des livres rares et anciens du Séminaire de Québec donne un accès privilégié aux sources à partir desquelles ont été formés les premiers lettrés québécois.

---

[1] Voir Henri-Jean Martin et Roger Chartier, *Histoire de l'édition française*, Paris, Promodis, 1983-1986, 4 vol.

[2] Voir Fiona Black et Yvan Lamonde (dir.), *Histoire du livre et de l'imprimé au Canada / History of the Book in Canada, II (1840-1918)*, Montréal et Toronto, Presses de l'Université de Montréal et University of Toronto Press, 2005.

Or, les humanités numériques ont, au cours des deux dernières décennies, transformé en profondeur ce domaine d'étude. Dans un premier temps, des analyses quantitatives qui s'intéressent à un patrimoine plus matériel ont été mises sur pied. Au Québec, c'est le cas du projet IMAQ (Inventaire des imprimés anciens au Québec) qui consiste à faire un vaste travail d'inventaire afin de recenser les imprimés européens des XVI^{e}, XVII^{e} et XVIII^{e} siècles présents sur le territoire québécois.

Dans un second temps, la recherche s'est intéressée à ce même patrimoine, mais de manière à en proposer des analyses plus qualitatives. L'objectif de ce chantier est de créer des outils d'animation intellectuelle et culturelle autour de bibliothèques patrimoniales québécoises disparues et des moyens de communications (applications, plateformes d'édition ou de diffusion). Mes travaux de recherche s'inscrivent dans ce contexte, plus précisément dans l'étude des *marginalia*, c'est-à-dire des notes manuscrites tracées dans les imprimés anciens. Ils se rattachent plus précisément au 3^{e} axe de la programmation scientifique de l'équipe Museum Numericum (MN 16-18) dont nous faisons partie. Cet axe concerne l'optimisation de leur base de données en ligne[3], et vise notamment à offrir un outil d'investigation plus performant aux équipes travaillant sur le livre ancien, de manière à faire en sorte que l'expertise acquise puisse aussi servir à des projets portant sur le patrimoine lettré du Québec moderne (bibliothèque d'Anne Hébert).

De fait, avec l'évolution des techniques d'apprentissage automatiques et d'apprentissage profond, plusieurs domaines d'activité demandent à avoir ces techniques intégrées à leur système pour faciliter certaines tâches qui prendraient généralement des jours, des mois, voire des années à être effectuées. Dans le domaine des livres anciens et des archives plus particulièrement, les chercheurs éprouvent de nombreuses difficultés dans leur exploitation. Les difficultés qu'ils rencontrent sont nombreuses, mais nous n'en énumérons que quelques-unes :

- Recherche de contenus textuels;

---

[3] Depuis http ://www.uqtr.ca/biblio, on consultera l'Outil de découverte; le catalogue « IMAQ – Livres anciens » est accessible via le menu déroulant de la fenêtre de recherche.

- Proposition d'archivage des pages des livres anciens;
- Détection des annotations;
- Transcription du contenu textuel.

Dans ce mémoire, nous nous sommes intéressés à la détection des annotations (et, de ce fait, à la classification des pages des documents) et à la transcription du contenu textuel des pages de livres.

Or plusieurs recherches ont été entreprises au cours des dernières années dans le domaine des *marginalia*. La plupart des travaux consistent à classifier les pixels des pages de documents contenant des annotations [1]. Dans notre travail, nous nous sommes intéressés au domaine de la classification de documents; plus explicitement, nous voulons développer un outil qui permet d'analyser des milliers de pages de livres anciens scannées, de les retranscrire et de déterminer la classe d'appartenance de chacune d'elles (imprimés des livres anciens) de manière automatique.

Pour atteindre cet objectif, nous nous sommes appuyés sur les réseaux de neurones convolutifs qui se sont montrés efficaces dans l'extraction de caractéristiques dans les images [2] pour catégoriser les pages, que nous avons distinguées en fonction des deux classes les suivantes : **(1) Annotée** pour celles comportant des *marginalia* ; et (2) **Non-annotée** pour les autres. Les réseaux de neurones convolutifs ont besoin d'un nombre conséquent de données étiquetées pour obtenir d'excellentes performances [3, 4, 5, 6] ; cependant, nous disposons d'un nombre limité de données.

Pour résoudre cette problématique, nous avons utilisé une technique d'augmentation de données et le modèle du nom de VGG16 [3], qui est un type de réseau de neurones convolutif comportant plusieurs couches de convolution et qui a fait ses preuves spécifiquement dans le domaine de la classification de documents [7, 8]. Nous l'avons pré-entrainé sur une base de données ImageNet [9] grâce à une autre technique appelée le "Transfer Learning" [10].

1.2 - Organisation

Ce mémoire est divisé en cinq grandes parties. La première partie traite de l'état de l'art; nous présenterons dans cette partie, différentes recherches qui ont été entreprises dans le domaine des annotations manuscrites, de la classification de documents, et du Transfer Learning. La deuxième partie sera consacrée au réseau de neurones artificiels et, avant d'aborder le cas particulier des réseaux de neurones convolutifs, nous explorons également différentes techniques qui peuvent être utilisées pour obtenir des résultats satisfaisants. Le chapitre 4 présente les outils, les méthodes et les données que nous avons utilisées pour atteindre l'objectif que nous nous sommes fixés dans ce travail. Puis, dans le chapitre 5, nous présenterons les résultats que nous avons obtenus et nous les interpréterons. Enfin, dans la dernière partie, nous ferons un résumé du mémoire de manière à conclure en présentant un bilan des résultats et les recherches qui seraient susceptibles d'être entreprises dans le domaine de la détection d'annotation manuscrites.

CHAPITRE 2. ÉTAT DE L'ART

Les annotations manuscrites et la classification de documents sont des domaines qui ont attiré l'attention de beaucoup de chercheurs [1, 7, 8, 10, 11, 12, 13, 14, 15, 16].

Les annotations manuscrites dans les livres anciens correspondent souvent à des notes, des commentaires, des explications, des marques de possession ou des critiques des lecteurs [1, 17]. Ces annotations donnent des indices sur l'histoire de la circulation d'un livre entre différents individus ou diverses institutions et, de ce fait, sur la circulation des savoirs. Elles nous renseignent aussi sur les pratiques de lecture, c'est-à-dire les manières de lire, en nous permettant notamment de comprendre comment les lecteurs ont utilisé leurs livres, les références qu'ils avaient en tête lorsqu'ils les lisaient, etc [17]. Elles peuvent recourir à des styles et des instruments d'écriture (stylo, crayon, surligneur) différents, être de couleurs différentes, mais aussi peuvent se trouver à divers endroits d'une page sous diverses formes (surlignage, soulignage ou par l'apposition d'un signe) [1, 17]. Leur complexité et leur diversité rendent la mise en place d'algorithmes permettant leur détection ou leur exploitation difficiles.

2.1 - Annotation Manuscrite et Segmentation Sémantique

Plusieurs travaux de recherches ont été entrepris dans le domaine des annotations manuscrites et de la classification des pixels. Lorsqu'on classifie des pixels d'une image ou qu'on associe une classe à des pixels; on parle de *Segmentation Sémantique*. Les approches traditionnelles dans ce domaine soutiennent qu'il est important qu'on puisse préliminairement localiser les régions dans les documents où se trouvent les caractéristiques et les annoter à la main, puis les fournir à un classificateur [1, 18, 19].

K. Zagoris *et al* [18] à l'aide d'un modèle appelé *Bag of Visual Words* ont réussi à extraire les caractéristiques locales de leur image pour les fournir ensuite à un classificateur SVM (*Séparateurs à vaste marge*), afin d'identifier et de séparer les annotations manuscrites du texte imprimé à la machine. Cette approche nécessite des phases de prétraitement qui dépendent du type de document.

K. Chen *et al.* [19] ont suggéré une approche basée sur le même classificateur (SVM) pour regrouper les pixels en deux classes : *« text line »* et *« non text line »*. Leur approche peut être utilisée seulement sur des images en couleur et ne nécessite pas des phases de prétraitement comme la précédente. Cependant, une étape de post-traitement basée sur un seuillage est nécessaire pour raffiner les résultats et peut dépendre des données d'apprentissage. Les images sont annotées à la main avant d'être fournies au classificateur. Ils ont procédé à la suppression des caractéristiques redondantes et non indispensables à l'aide de l'algorithme FCBF (*Fast Correlation-Based Filter*). Ils ont obtenu une précision moyenne de 91% et un rappel de 84% sur trois ensembles de données d'images manuscrites historiques de nature diverse.

Les travaux les plus récents dans le domaine de la segmentation des documents historiques s'appuient sur l'utilisation des modèles d'apprentissage profond.

K. Chen *et al.* [20] par exemple ont utilisé des auto-encodeurs à trois niveaux pour extraire les caractéristiques automatiquement à partir de l'intensité des pixels sans la nécessité d'annoter les pixels à la main. Ils ont utilisé ensuite ces caractéristiques pour entraîner un SVM afin de prédire la classe d'appartenance de chaque pixel. Leur méthode a été évaluée sur trois bases de données *(George Washington, Parzival, Saint Gall).* En obtenant un taux d'exactitude de 92.65% et de 96.64% sur les deux premières bases de données, ils ont démontré ainsi qu'il est possible d'obtenir un haut niveau de performances sans annoter les pages à la main [20].

D'autres modèles d'apprentissage profond supervisé s'appuyant sur les FCNN (*Fully Connected neural Network*) ont été proposés.

K. Andreas *et al.* [1] ont effectué une tâche de segmentation sémantique sur un jeu de données de 50 images annotées à la main (supervisé) à l'aide d'un Ground Truth pour différencier les pixels comportant les annotations de ceux n'en comportant pas. Les auteurs ont appelé la classe des pixels comportant des annotations manuscrites *« Handwritten annotation »* et la classe des pixels n'en comportant pas *« background »*. Ils ont effectué une série d'expérimentations avec différents modèles essentiellement des FCN-8s, le DeepLab V2 et PSPNet et différentes techniques de pré-entraînement et

d'augmentation de données. Ces modèles sont basés sur des architectures de réseaux de neurones convolutifs telles que le ResNet-101, ResNet-50 [5] le VGG16 [3]. Au terme de leurs travaux, ils ont obtenu avec le FCN-8s (modèle basé sur le VGG16 [3]) pré-entrainé sur ImageNet en utilisant l'*inception* comme technique d'augmentation de donnée un *mean IoU (Intersection over Union)* de 95.6% sur les données de test. Ils ont démontré que les approches qui se basent sur les FCNN sont capables d'effectuer des tâches de segmentation de documents efficacement [1].

2.2 - Classification de documents

On retrouve, dans la littérature, différentes approches pour classifier des documents ou dans la recherche de documents dans des bases de données.

Les approches les plus communément proposées exploitent les similarités dans la mise en page ou dans la structure des documents (c'est le cas de C. Shin et D. Doermann [21] et Y. Byun et Y. Lee [22]) ou exploitent le contenu textuel des documents en utilisant des techniques telles que la reconnaissance optique de caractères (OCR) [23]. D'autres combinent ces deux approches (Exploitation du contenu textuel et de la structure des documents) [24].

Les approches les plus récentes se basent sur les techniques d'apprentissage profond, plus particulièrement les réseaux de neurones convolutifs pour l'extraction de caractéristiques et pour la classification des documents.

L'un des travaux pionniers dans ce domaine est celui de Le Kang et al [14]; grâce au réseau de neurones convolutif, les auteurs ont démontré qu'il est possible d'extraire les caractéristiques de manière automatique pour la classification de documents. Leur étude s'est faite sur la base de données *Tobecco-3481* [25] qui contient 3482 images de documents catégorisées en 10 classes (*report, memo, resume, scientific, letter, news, note, ad, form, email*) et sur la base de données NIST [26] qui comprend 5590 images de documents (des formulaires d'impôt) catégorisées en 20 classes. Ils ont obtenu un taux d'exactitude moyen de 65.37% sur la première base de données et de 100% pour la seconde base de données.

L'introduction de la base de données RVL-CDIP [27] qui comprend 400 000 documents classifiés en 16 catégories a permis l'évaluation des CNN sur un nombre conséquent de données. C'est ainsi que C. Tensmeyer et T. Martinez [15] ont évalué les performances de leur réseau de neurones convolutif sur cette base de données pour une tâche de classification de documents. Ils ont réussi à obtenir des performances de 90.8 % en utilisant des images en entrée de grandes tailles et en appliquant le *shear transform* comme technique d'augmentation de données sur les données d'entraînement [15].

L'entraînement des réseaux de neurones convolutifs sur de grandes bases de données a montré qu'il est possible de transférer les connaissances ou l'expérience acquise sur ces dernières et l'appliquer à d'autres domaines ou tâches similaires [28, 29, 30] : on parle d'apprentissage par transfert (*Transfer Learning).* C'est avec l'avènement des grandes bases de données telles que ImageNet [9] et CIFAR-10 [31] que son efficacité a été démontrée bien que cette technique existait depuis environ deux décennies [32].

M.Z Afzal *et al.* [10] nous montre qu'il est possible de transférer les connaissances ou caractéristiques acquises sur un problème de classification général tel que ImageNet [9] et de l'appliquer à un problème de classification de documents. Ils ont obtenu un taux d'exactitude moyen de 77.6 % sur les bases de données Tobecco-3481 [25] et NIST [26] en utilisant cette technique [28].

M.Z Afzal *et al.* [7] ont réussi à obtenir un taux d'exactitude de 90.97% grâce au VGG16 [3] pré-entrainé sur ImageNet [9] dans le but de classifier les documents de la base de données RVL-CDIP [27].

À cette même fin, Arindam Das *et al.* [8] dans leur article ont présenté une approche basée sur le VGG16 [3] et sur le *Transfer Learning* à deux niveaux. Ces deux niveaux de *Transfer Learning* sont appelés 'inter-domain' et 'intra-domain'. Le Transfer Learning 'inter-domain' consiste à transférer les poids du VGG16 [3] pré-entrainés sur la base de données ImageNet [9] pour entraîner un classificateur de documents sur les images entières de documents qu'ils ont appelé « Holistic ». Le deuxième niveau de *Transfer Learning* 'intra-domain' consiste à transférer les poids du précédent modèle

(VGG16 [3] pré-entrainé sur les images entières des documents) pour entraîner un autre classificateur sur différentes régions des documents : l'entête, le pied de page, et les régions de droites et gauches de la page (*Header, Footer, Left and Right Body*). Ce deuxième niveau de *Transfer Learning* permet un apprentissage plus rapide. Enfin, ils ont combiné les prédictions des classificateurs basées sur les régions et celles basées sur les images entières de documents à l'aide des modèles d'apprentissage profond. Leur méthode a atteint un taux d'exactitude de 92,2%.

2.3 - Conclusion

Parmi les approches qui ont été proposées dans le domaine de la classification des documents, celles qui utilisent les réseaux de neurones convolutifs se sont montrées les plus prometteuses. Le VGG16 [3] est un modèle qui a fait ses preuves non seulement dans le domaine de la segmentation sémantique des annotations manuscrites [1], mais aussi dans le domaine de la classification de documents [7, 8]. Pour arriver à ces performances, les auteurs ont pré-entrainé ce modèle sur la base de données ImageNet [9] dont les poids sont publiquement disponibles grâce au *Transfer Learning* qui est une technique permettant le transfert de connaissances.

Nous allons donc utiliser ce modèle et le pré-entraîner sur cette base de données, puis nous l'adapterons à notre problème et notre jeu de données pour classifier nos différentes pages.

CHAPITRE 3. RÉSEAU DE NEURONES CONVOLUTIFS

3.1 - Introduction

Nous allons présenter, dans ce chapitre, l'architecture des réseaux de neurones artificiels et des réseaux de neurones convolutifs et, en particulier, le VGG16 [3] qui est un réseau de neurones convolutif éprouvé dans le domaine de la classification de documents. Nous présenterons aussi différentes techniques qui peuvent être utilisées pour améliorer les performances de nos différents modèles.

3.2 - Principes des réseaux de neurones artificiels

Un réseau de neurones artificiels est un système dont la conception s'inspire du fonctionnement du cerveau et du système nerveux humain. Il existe plusieurs types de réseaux de neurones dont le plus populaire est le perceptron multicouche.

Le perceptron multicouche est introduit en 1957 par Frank Rosenblatt [33]. Nous vous présenterons par la suite son architecture et son fonctionnement.

3.2.1 - Architecture et fonctionnement des réseaux à couches

Un réseau de neurones artificiels ou un perceptron multicouche est un réseau composé d'une succession de couches; généralement de trois couches. Ces couches sont les suivantes (voir figure 1) :

- La couche d'entrée : qui est la couche à laquelle nos données brutes sont présentées. Nos données sont généralement présentées sous forme de vecteurs. Elle reçoit les informations censées expliquer le phénomène à analyser [34];
- La/ les couche(s) cachée(s) : est la couche qui se trouve entre la couche d'entrée et celle de sortie. Il s'agit du cœur du perceptron servant à effectuer des calculs intermédiaires [34]. Un réseau peut avoir une ou plusieurs couches cachées;

- La couche de sortie : est la dernière couche d'un réseau de neurones; celle qui donne le résultat du calcul interne [34].

NB : Un perceptron peut être constitué de couches d'entrée et de sortie uniquement.

Chacune de ces couches contient des nœuds, appelés neurones, qui se comportent comme des unités de calcul autonome [34]. Les cercles 1-9 présentés à la figure 1 représentent les neurones.

La couche d'entrée de notre figure comprend 5 neurones, la couche cachée comprend 3 neurones et la couche de sortie 1 neurone [34]. Les neurones d'une même couche n'ont aucune connexion entre eux.

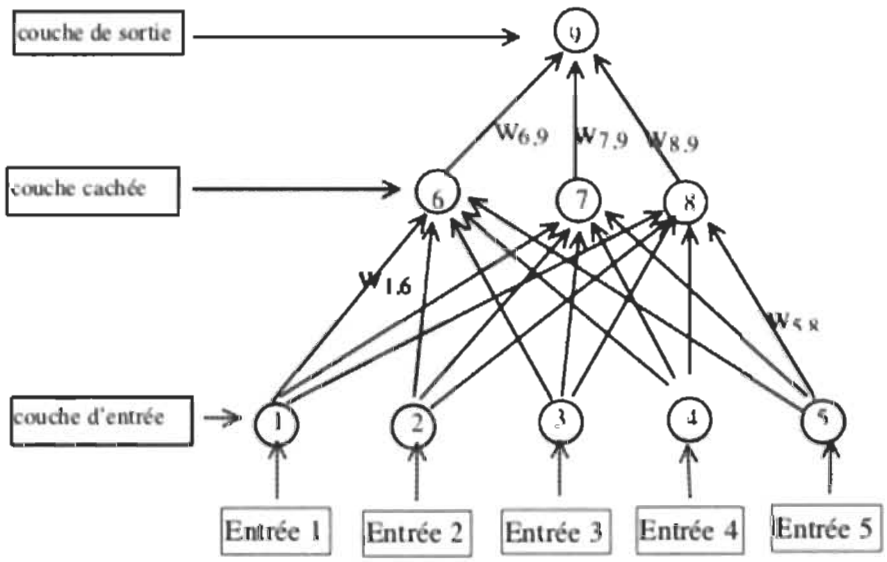

Figure 1 - Architecture d'un réseau à couche [34].

Les réseaux de neurones font partie de la grande famille des techniques d'apprentissage automatique et supportent les deux types d'apprentissage que sont l'apprentissage supervisé et l'apprentissage non-supervisé :

- Supervisé : Lors de l'entraînement, des paires de données d'entrée et de sortie (des données étiquetées) sont présentées au réseau. Ce dernier apprend et se paramètre à

partir de ces paires de données dans le but d'obtenir les étiquettes souhaitées [35]. L'apprentissage supervisé peut donc être utilisé lorsque nous disposons de données étiquetées. Lorsque les données à prédire en sortie sont des catégories ou des classes, il s'agit d'un problème de classification. Dans le cas où ces données sont des valeurs quantitatives (par exemple des salaires) on parle de problème de régression.

- Non-supervisé : Dans ce type d'apprentissage, les réseaux reçoivent en entrée des données non étiquetées c'est-à-dire des données dont les sorties ne sont pas connues au préalable par le réseau. Ce dernier va par lui-même découvrir des similarités ou récurrences ou les structures cachées dans les données et converger vers n'importe quel état final [35, 36].

Notre problématique sera abordée par l'apprentissage supervisé ce qui va nécessiter un entraînement sur des données étiquetées.

3.2.2 - Fonction d'activation

Les neurones de chaque couche sont interconnectés aux neurones des autres couches par des liaisons appelées poids synaptiques notés Wi, j (voir figure 1).

Ces poids sont initialisés de manière aléatoire et déterminent la quantité d'influence de la sortie d'un neurone sur un autre neurone. Celle-ci est calculée en faisant la somme des potentiels de sortie de ses prédécesseurs pondérée par les poids synaptiques (noté A sur notre figure 2). Le résultat de ce calcul est transformé par une fonction (notée S sur la figure 2) appelée fonction d'activation qui le ramène à l'intérieur d'une borne définie. Cette fonction est appliquée uniquement à la sortie des neurones de la ou des couches cachées [34, 37]

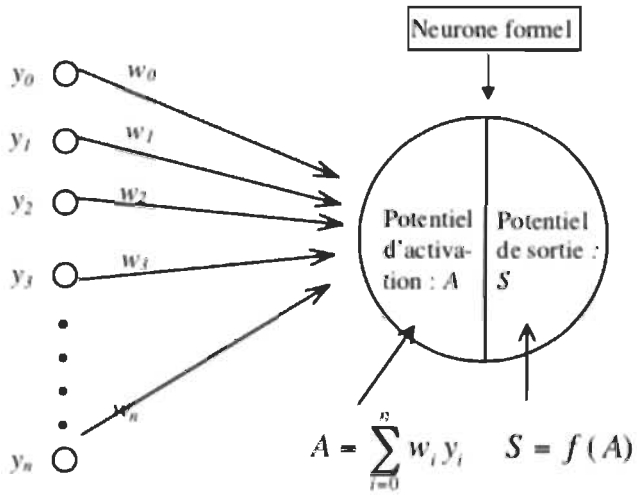

Figure 2 - Schéma d'un neurone artificiel [34].

Il existe plusieurs types de fonction d'activation, mais la plus populaire est la fonction *Relu* (Rectification Linéaire), qui est la plus souvent utilisée au niveau des couches cachées, et la fonction sigmoïde appliquée à la couche de sortie pour un problème de classification binaire.

L'expression analytique de la fonction *Relu* est : $f(A) = \max(0, A)$.

L'expression analytique de la fonction Sigmoïde est : $f(A) = \frac{1}{1 + e^{-A}}$

3.2.3 - Algorithme d'apprentissage et fonction de perte

Dans le cas d'un apprentissage supervisé, des paires de données formées d'observations et d'étiquettes sont fournies au réseau. Dans chaque paire, l'étiquette représente le résultat (classe) désiré pour l'observation associée en entrée.

L'algorithme d'apprentissage a pour tâche d'évaluer et d'ajuster les poids du réseau en fonction des données présentées en entrée de sorte à obtenir les résultats souhaités en sortie car la modification des poids d'un neurone correspondant influe sur

tous les neurones de la couche suivante et sur les résultats du neurone de la dernière couche [34, 38, 39].

Lors du passage des données de la couche d'entrée à la couche de sortie; le neurone de la dernière couche (la couche de sortie) donne un résultat qui correspond à la prédiction de notre réseau.

Une erreur est ensuite estimée à l'aide d'une fonction (appelée fonction de perte ou de coût) qui détermine l'écart qu'il y'a entre la sortie prédite et la sortie réelle. L'objectif est de minimiser l'erreur [34, 38, 39]. Il existe plusieurs types de fonction de coût, mais celle qui nous intéresse est la fonction de perte entropie couramment utilisée pour des problèmes de classification (lorsque la sortie est une catégorie).

3.2.4 - La rétropropagation et les algorithmes d'optimisation

L'erreur estimée au moyen de la fonction coût est rétro-propagée de la couche de sortie vers la couche d'entrée et les poids synaptiques sont modifiés ce qui permet d'initier un nouveau calcul. Ce processus est appelé rétropropagation. Il se poursuit ainsi jusqu'à ce que l'utilisateur intervienne pour y mettre fin ou jusqu'à ce qu'une valeur de la fonction coût ou un seuil fixé a priori est atteint [34, 38].

La fonction coût est minimisée par l'algorithme de descente du gradient qui est un algorithme d'optimisation itératif qui procède à l'amélioration successive de l'erreur en prenant en compte les modifications antérieures des poids synaptiques qui ont contribué de manière significative à celle-ci [34, 38]. L'objectif est de diriger la fonction de perte à petit pas vers un minimum global (figure 3).

Le calcul du gradient peut se faire de différentes manières. On peut faire passer l'ensemble de nos données d'apprentissage dans notre réseau, puis calculer l'erreur global c'est-à-dire l'erreur sur l'ensemble de ces données : on parle de batch.

Ce calcul peut mener à un problème de minimum local, comme c'est le cas à la figure 4, ou être très excessif. C'est pourquoi l'algorithme de gradient stochastique ou en

ligne a été mis en œuvre. Il consiste à mettre à jour les paramètres ou poids pour chaque exemple ou échantillon choisi aléatoirement dans l'ensemble d'entraînement.

L'erreur locale, c'est-à-dire l'erreur pour chaque échantillon de notre ensemble d'apprentissage, est minimisée contrairement à la première approche (batch) où l'erreur globale est minimisée [40, 41]. On rencontre peu de problèmes dus aux minima locaux en raison du fait que l'erreur est minimisée localement. Plusieurs parcours de l'ensemble de nos exemples peuvent être nécessaires avant la convergence de cette méthode [38].

La méthode de descente de gradient stochastique avec moment est une approche qui s'est aussi montrée efficace. Sa particularité est qu'elle conserve en mémoire à chaque itération les changements précédents des poids, ce qui lui permet d'éviter les oscillations et d'échapper au minimum local mais aussi d'accélérer l'optimisation du réseau [42].

Une autre approche ne consiste ni à calculer l'erreur pour chaque exemple ni à calculer l'erreur de l'ensemble des exemples d'apprentissage, mais l'erreur d'un groupe ou d'un ensemble d'exemples dans notre ensemble d'apprentissage. Cette approche est appelée « mini-lots ». Elle peut se montrer efficace, car elle réduit la variance de nos poids lorsqu'ils sont réévalués et permet un traitement parallèle des observations, mais elle permet aussi une convergence plus lisse et plus stable [43].

Il convient à l'utilisateur de déterminer l'architecture optimale du réseau, afin d'obtenir une bonne prédiction [38].

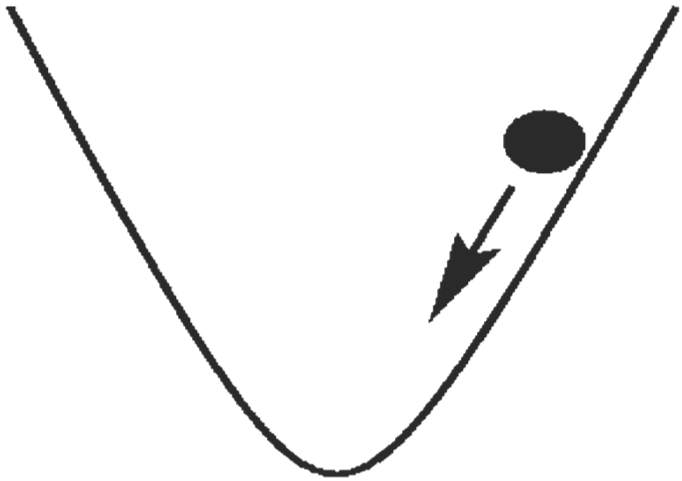

Figure 3 - Illustration Minimum global [44].

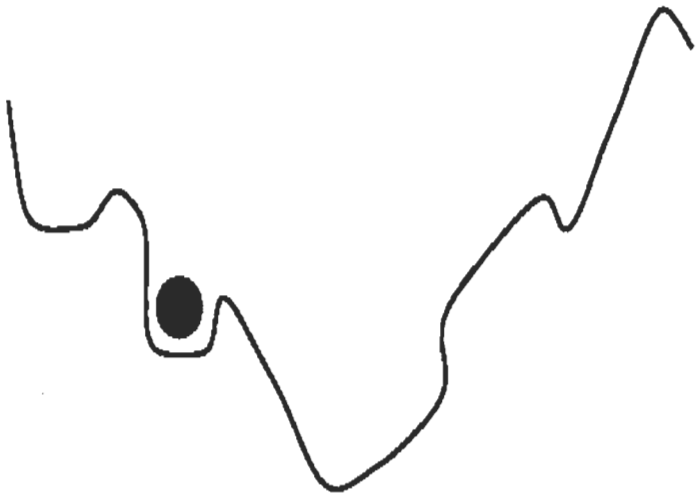

Figure 4 - Illustration minimum local [44].

3.2.5 - Données d'entraînement, de validation et de test

La mise en œuvre d'un réseau nécessite la répartition des données en 3 échantillons qui sont l'échantillon d'entraînement, de validation et de test.

L'échantillon d'entraînement et l'échantillon de validation sont utilisés lors de la phase d'entraînement de notre réseau. L'échantillon d'entraînement est utilisé pour

ajuster les poids du modèle dans le but de minimiser la fonction de perte et d'obtenir les résultats souhaités en sortie. Le modèle est également soumis à un ensemble de validation qui est différent de l'échantillon d'entraînement et qui sert à valider notre modèle. Ce dernier sert à déterminer la capacité de généralisation du modèle.

Pour obtenir une bonne capacité de généralisation, un modèle doit donner de bons résultats aussi bien sur l'ensemble d'entraînement que sur l'ensemble de validation. Il convient donc à l'utilisateur d'arrêter l'apprentissage lorsque le modèle atteint cet objectif (voir figure 5).

L'échantillon de test, quant à lui, sert à déterminer les performances du modèle sur de nouvelles données, c'est-à-dire des données que le modèle n'a pas encore vues [34].

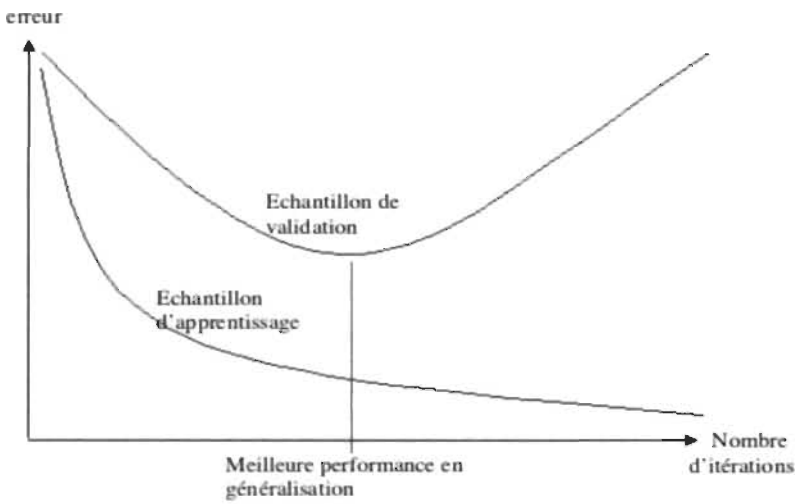

Figure 5 - Évolution de la courbe d'erreur durant la phase d'apprentissage [34].

3.2.6 - Quelques notions

On parle de « surapprentissage » lorsque le modèle donne de bons résultats sur nos données d'entraînement, mais de piètres résultats sur nos données de validation.

Le terme itération *« epoch »* détermine le nombre d'itération sur l'ensemble de nos données d'entraînement.

3.3 - Réseau de neurones convolutifs

Les réseaux de neurones convolutifs sont des réseaux couramment utilisés pour des problèmes de classification d'images; ces derniers se sont montrés efficaces dans divers travaux de recherche grâce à leur capacité à extraire des caractéristiques dans les images [2].

La structure d'un réseau de neurones convolutif (comme le montre la figure 6) est essentiellement composée de couche(s) de convolution, de couche(s) de Pooling et de couche(s) d'un réseau de neurones artificiels entièrement connectée [37]. La couche principale et la plus importante de ce type de réseau est la couche de convolution d'où son appellation : réseau de neurones convolutif.

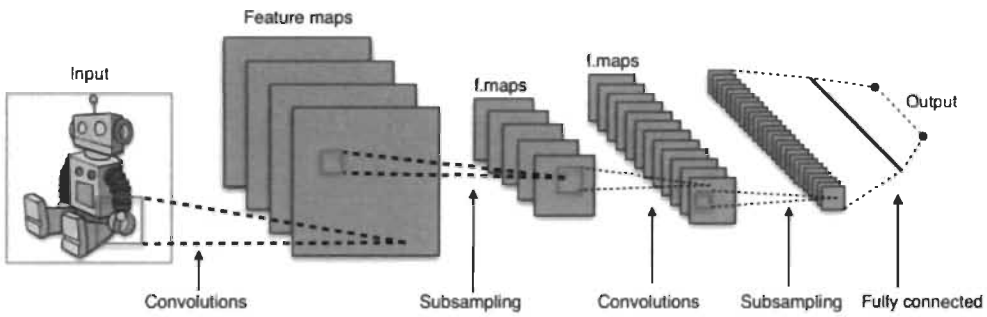

Figure 6 - Structure d'un réseau de neurones convolutifs [45].

3.3.1 - Images RVB

Les images sont généralement sous la forme RVB (Rouge -Vert – Bleu), c'est-à-dire qu'elles sont représentées sous trois filtres de couleurs de même dimension. Chacun de ces filtres est représenté sous forme matricielle et prend des valeurs appelées pixels qui donnent l'intensité du rouge, du vert et du bleu.

Ces valeurs sont comprises entre 0 et 255 où 0 correspond à la couleur noire pure et 255 correspond au blanc pur. Les lignes représentent la largeur de l'image, les colonnes correspondent à la hauteur de l'image et le nombre de filtres correspond à la profondeur de l'image. La figure 7 par exemple est la représentation d'une image RVB de taille 4x4x3 qui a une largeur et une hauteur égale à 4 et une profondeur égale à 3.

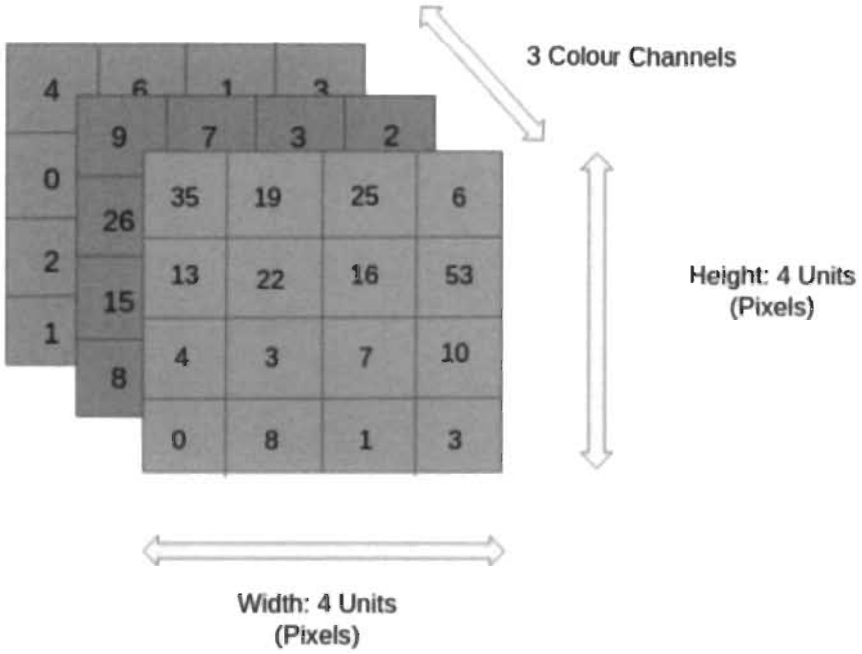

Figure 7 - Image RVB représentée sous forme matricielle [46].

3.3.2 - La couche de convolution

La couche de convolution est la première couche d'un réseau de neurones convolutif. Un réseau de neurones convolutif peut être constitué d'une ou de plusieurs couches de convolution. Cette couche fonctionne comme un extracteur de caractéristiques; elle prend en entrée des images de type RVB [47].

L'opération de convolution est la somme de la multiplication élément par élément de filtres ou noyau de convolution par de petites zones ou tranches de notre matrice d'image d'entrée (voir figure 8). Ces petites zones de notre image d'entrée sont de même taille que le filtre. Le filtre chevauche notre image d'entrée jusqu'à ce qu'il parcoure tous les pixels de cette dernière, comme l'illustre la figure 9.

Le filtre ou noyau de convolution a une largeur, une hauteur et une profondeur, la profondeur déterminant le nombre de filtres et le nombre d'images en sortie. C'est à l'utilisateur de déterminer la taille du filtre. Les valeurs ou poids de ces filtres sont initialisés aléatoirement et constituent les paramètres qui seront appris lors de la phase d'apprentissage [47, 48]. Par exemple, une couche convolutive ayant un filtre de taille

3x3x64 signifie que nous avons 64 filtres de largeur et de hauteur égale à 3. Cela signifie aussi que nous obtenons en sortie 64 images.

Le filtre chevauche notre image d'entrée en faisant des pas. Un pas (*stride*) est un paramètre essentiel lors d'une opération de convolution [48]. En effet, plus la valeur du pas est grande plus la taille de l'image en sortie est petite.

À la suite de l'opération de convolution (du chevauchement de l'image par le filtre) on obtient en sortie de nouvelles images appelées des cartes de convolution *(Feature Maps)* qui sont des images réduites de l'image originale et faisant ressortir des caractéristiques distinctives de celle-ci [37, 47, 48]. Plus nous avons de couches convolutives, plus la taille de l'image originale tend à être réduite. C'est pour cette raison qu'une méthode appelée la marge à zéro (zero padding) est souvent utilisée. Cette dernière consiste à ajouter des pixels supplémentaires aux images d'entrée des couches convolutives, de manière à obtenir des images de mêmes tailles en sortie [48].

Les premières couches convolutives d'un réseau de neurones convolutif révèlent des caractéristiques primaires, mais les dernières arrivent à obtenir des images en sortie présentant des caractéristiques plus abstraites ou cachées [47].

Une fonction d'activation enfin est introduite aux sorties (cartes de convolution) de chaque couche de convolution. Cette dernière permet d'introduire une non-linéarité au système. La fonction *Relu* est la fonction d'activation la plus utilisée, car elle permet un entraînement plus rapide sans dégrader les performances du réseau [4, 49].

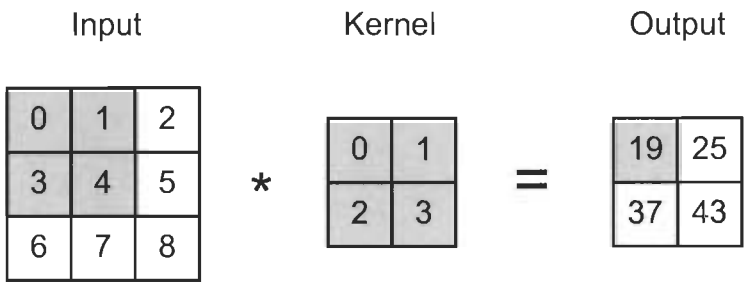

Figure 8 - Opération de convolution [50].

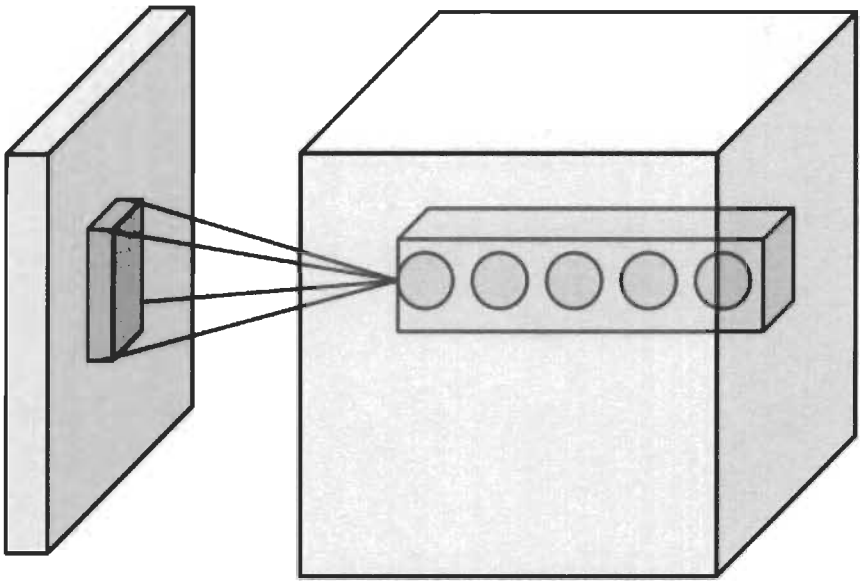

Figure 9 - Opération de convolution [51].

3.3.3 - La couche de Pooling

Les couches de convolution sont souvent suivies de couches de Pooling. Ces couches effectuent des opérations sur les cartes de convolution que nous obtenons en sortie des couches de convolution. Elle divise chacune des cartes de convolution en tranches qui sont des matrices de mêmes tailles et, effectue des pas pour parcourir l'ensemble des tranches.

La taille de ces matrices et des pas est déterminée par l'utilisateur. Généralement des tranches de petites tailles sont utilisées (par exemple 2x2 ou 1x1). Des tranches de trop grandes tailles diminueraient significativement la taille de nos cartes de convolution et il en résulterait une perte énorme d'informations.

Ensuite on prend la valeur maximale ou moyenne de chacune des tranches (matrices). Lorsque celle-ci prend la moyenne des matrices, on parle de *« Average Pooling »* ; lorsque c'est la valeur maximale (activation forte) qui est prise, on parle de *« MaxPooling »* [47]. La dernière *« MaxPooling »* est l'opération de Pooling la plus utilisée. La figure 10 nous montre l'application de l'opération *« MaxPooling »* sur une

matrice d'image de tailles 4x4 où la tranche est de taille 2x2 avec le pas de taille 2. La valeur maximale de la première tranche de cette matrice qui est en orange est 6. Pour les autres tranches, nous obtenons comme valeur maximale successivement 8, 3 et 4. On obtient à la suite de cette opération une image de taille 2x2.

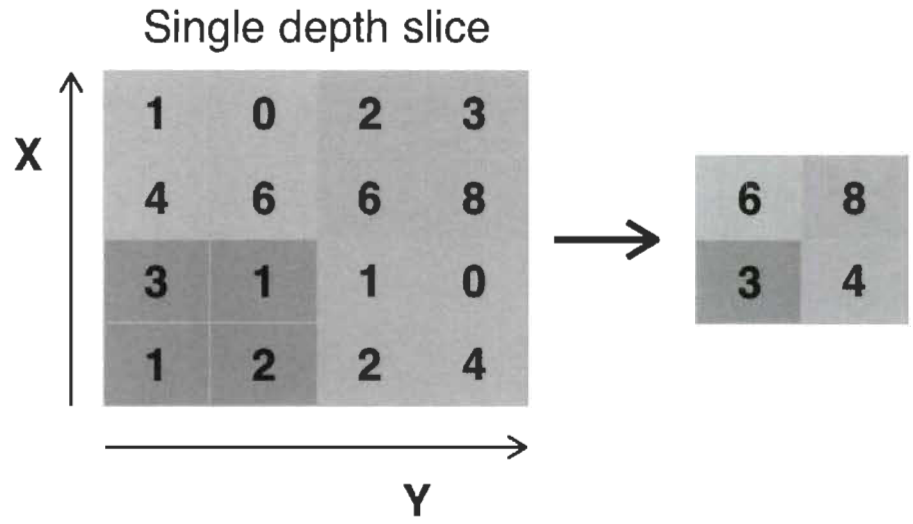

Figure 10 - MaxPooling [52].

L'objectif de cette méthode est de réduire la dimension de chaque carte de convolution c'est-à-dire la largeur et la hauteur de chacune d'elle réduisant ainsi le nombre de paramètres de notre réseau de neurones convolutif et le coût computationnel de celui-ci.

L'opération de Pooling élimine les informations non nécessaires ou les bruits contenus dans nos cartes de convolution. Elle rend également notre réseau plus robuste aux changements d'orientation et aux positions des caractéristiques dans nos images; elle crée une forme d'invariance par translation [47, 48].

3.3.4 - L'opération Flattening et la couche entièrement connectée

Les sorties (les matrices) que nous obtenons à la suite d'une série d'opérations de convolution et de Pooling, sont transformées en vecteur : cette opération est appelée

Flattening. Elle consiste à regrouper toutes les caractéristiques de notre image extraites en un seul vecteur (voir figure 11).

Nous appliquons ensuite ce vecteur à un perceptron multicouche constitué de couches entièrement connectées afin de déterminer la classe d'appartenance de l'image en sortie [47, 48].

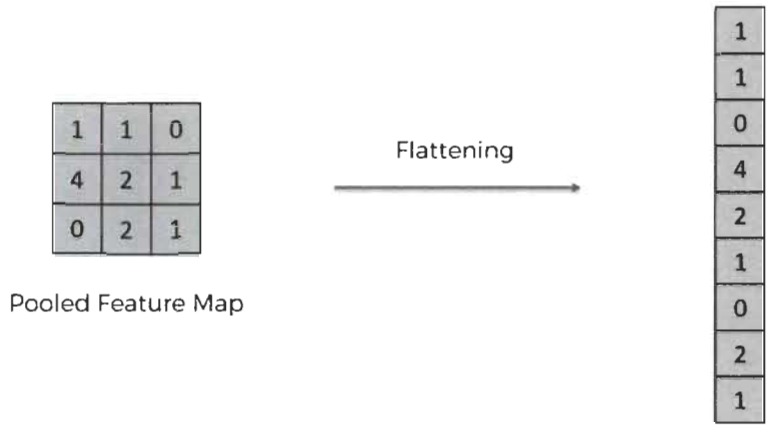

Figure 11 - L'opération Flattening [53].

3.3.5 - L'avantage des réseaux de neurones convolutifs

L'un des avantages majeurs des réseaux de neurones convolutifs est le partage de poids aussi appelé *« shared weight »* [4, 54]. En effet, pour une couche de convolution donnée, chaque neurone (filtre) de cette couche partage les mêmes poids. Cette caractéristique permet aux réseaux de neurones convolutifs de réduire le nombre de leurs paramètres, l'espace mémoire et d'améliorer la vitesse d'apprentissage [48].

Le perceptron multicouche au contraire associe un poids différent à chaque signal entrant aux neurones augmentant le nombre de paramètres et le coût computationnel [48]; dans ce cas, plus l'image est grande et plus le nombre de paramètres augmente.

Un autre avantage du réseau de neurones convolutif est qu'il est spatialement invariant, ce qui veut dire qu'il n'est pas sensible à la position d'une caractéristique dans

une image et peut détecter une caractéristique dans une image quelle que soit son orientation ou sa position, ce qui n'est pas le cas du perceptron multicouche [47].

3.4 - VGG16

Le VGG16 [3] est un type de réseaux de neurones convolutifs qui comprend plusieurs blocks de convolution. Il en existe plusieurs autres (LeNet, AlexNet, ResNet-50, GoogleNet), mais comparé à ceux-ci le VGG16 [3] a obtenu les meilleures performances pour les tâches de localisation de la compétition annuelle ILSVRC (*ImageNet Large-Scale Visual Recognition Challenge*) de 2014 [55]. Il a également obtenu les meilleures performances sur la base de données RVL-CDIP [7].

Ce modèle prend en entrée des images de tailles (224x224x3). Il est constitué de cinq blocks de couches de convolution (voir figure 12) accompagnés de couches de *MaxPooling*. Un filtre de taille 3x3 et un pas (*stride*) de taille 1 [3] sont utilisés pour chaque couche de convolution. La fonction d'activation *Relu* est utilisée pour chaque couche de convolution et la technique du zéro padding, de manière à conserver la taille de nos images après chaque couche de convolution.

Nos deux premiers blocks de convolution sont constitués chacun de deux couches de convolution et ont une profondeur de 64 et 128 successivement (figure 12). Nos trois derniers ensembles de convolution comprennent chacun trois couches de convolutions et ont successivement des profondeurs de 256, 512 et 512.

L'opération de Flattening est utilisée à la suite du dernier block de couches de convolution. Ensuite nous avons le classificateur qui comprend 3 couches (deux couches cachées de 4096 neurones chacune et une couche de sortie de 1000 neurones). La dernière couche comporte 1000 neurones, car ce modèle a été conçu pour la classification d'images en 1000 catégories [3, 55].

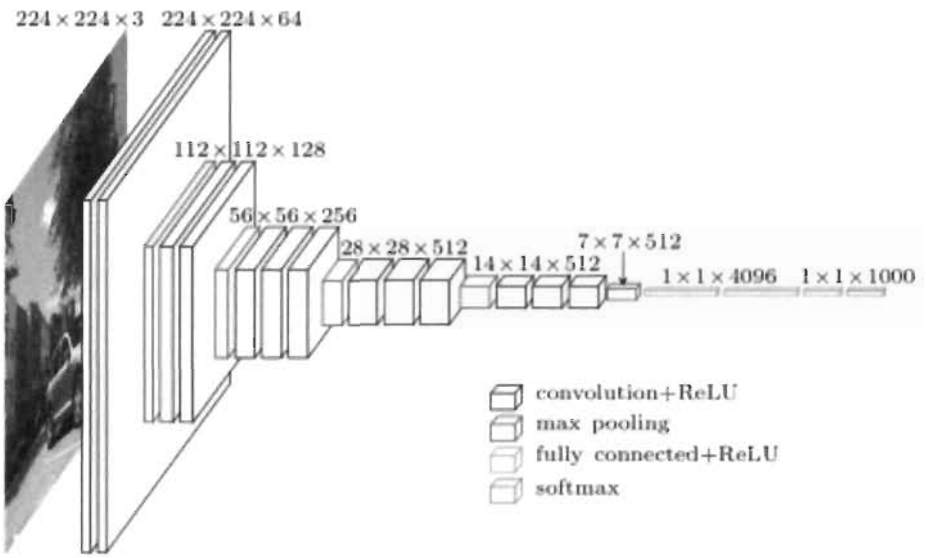

Figure 12 - Architecture du VGG16 [56].

3.5 - Amélioration de la précision

3.5.1 - Pré-entraînement ImageNet

ImageNet [9] est une base de données d'images destinée aux travaux de recherche dans le domaine de la vision par ordinateur.

Cette base de données comporte plus de 15 millions d'images qui appartiennent à 22.000 classes ou catégories étiquetées à la main et recueillies à partir de diverses sources sur le web [4].

Lancée en 2010, ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) est une compétition annuelle qui a pour but d'obtenir la meilleure prédiction sur plusieurs tâches dans le domaine de la vision par ordinateur tel que la détection et la classification d'objets dans les images naturelles. Cette compétition est basée sur un sous-ensemble d'ImageNet contenant 1,2 million d'images d'entraînement, 50 000 images de validation et 150 000 images de test classées en 1000 catégories [4].

Plusieurs modèles ont été pré-entrainés sur ce sous-ensemble à l'aide de la technique Transfer Learning pour exploiter les connaissances acquises sur celle-ci, et

plusieurs recherches ont démontré l'efficacité de pré-entraîner des modèles sur cette base de données [57, 58]. C'est le cas de [7] où les modèles AlexNet, ResNet-50, GoogleNet et VGG16 [3] (le modèle que nous avons décrit dans la section 3.4) ont été pré-entrainés sur ImageNet [9]. Ils sont arrivés à la conclusion, au terme de leurs recherches pour tous les modèles et après toutes leurs expérimentations, que les modèles fournissent de meilleurs résultats lorsqu'ils sont pré-entrainés sur ImageNet [9] que lorsqu'ils ne le sont pas [7].

3.5.2 - Taux d'apprentissage

Le Taux d'apprentissage (Learning Rate) est une valeur scalaire que les algorithmes d'optimisation tels que la descente de gradient utilisent pour déterminer la taille du pas à chaque itération tout en se déplaçant vers un minimum d'une fonction de perte [59]. C'est un hyper-paramètre qui est ajustable et qui peut influencer les résultats de notre modèle d'apprentissage. Il représente métaphoriquement la vitesse à laquelle un modèle d'apprentissage automatique « apprend » [60].

La nécessité de fixer un taux d'apprentissage est souvent problématique, car un taux d'apprentissage trop élevé fera sauter l'apprentissage au-dessus des minima, mais un taux d'apprentissage trop faible mettra trop de temps à converger ou restera coincé dans un minimum local indésirable [61]. N. Smith [62] montre (par un graphe comparant le taux d'apprentissage à l'erreur) que pour un taux d'apprentissage qui augmente, il existe une limite à partir de laquelle l'erreur arrête de diminuer et commence à augmenter.

3.5.3 - Taille de lot

La taille de lot *« Batch Size »* est généralement utilisée pour se référer à la taille d'un mini-lot ou mini-batch [63]. La taille d'un mini-lot est comprise entre 1 et le nombre total d'exemples de nos données. Par exemple pour un jeu de donnée de 100 images; la taille du mini-lot peut être soit 2, 4, 8, 64 ou n'importe quelle autre valeur comprise entre 1 et 100. Celle-ci influence la dynamique de l'algorithme d'apprentissage et elle correspond au nombre d'exemples de nos données après lesquels l'estimation du gradient d'erreur, les poids synaptiques et les paramètres sont recalculés et mis à jour.

La taille du lot a un impact sur les performances du modèle; dans les deux sections qui suivent nous allons présenter les avantages d'utiliser des lots de petites tailles et ceux de grandes tailles. À cette fin, Yoshua Bengio [64] affirme qu'il y'a une tension entre la taille des lots et la vitesse et la stabilité du processus d'apprentissage.

3.5.3.1 - Lot ou Batch de petite taille

Il a été démontré dans plusieurs recherches que les Lots de petite taille facilitent l'estimation du gradient d'erreur et permettent d'avoir un entraînement plus stable et plus fiable [65]. Les résultats de Dominic Masters et Carlo Lushi [65] sur la base de données ImageNet confirment que l'utilisation d'un lot de petite taille fournit des avantages au niveau des performances, mais aussi une convergence stable de l'apprentissage pour différentes valeurs du taux d'apprentissage.

Yoshua Bengio [64] démontre que les lots de petites tailles permettent d'obtenir de meilleures performances en termes de généralisation des données de test [66, 67, 68] mais aussi, il donne des valeurs pour lesquelles la taille du batch donne de bons résultats. Ces valeurs sont généralement comprises entre 32 et 2.

3.5.3.2 - Lot ou Batch de grande taille

Selon [68, 69, 70, 71, 72, 73], les lots de grande taille permettent le parallélisme sur plusieurs machines, réduisant les coûts computationnels et ainsi le temps d'apprentissage. Mais cela à une incidence sur les performances des données de test.

Les résultats de Dominic Masters et Carlo Lushi [65] sur les bases de données CIFAR-10, CIFAR-100 [31] et ImageNet [9] confirment qu'augmenter la taille des lots diminuerait la plage des taux d'apprentissage pour lesquels notre modèle fournit une convergence stable et des performances de test acceptable.

E. Hoffer *et al.* [74] ont fourni des résultats probants établissant qu'il est possible d'obtenir de meilleures performances de généralisation en élargissant le temps d'entraînement et en utilisant des lots de grandes tailles avec des taux d'apprentissage plus grands basés sur une règle d'échelle linéaire pour éviter les coûts computationnels.

3.5.4 - Dropout

Le « *Dropout* » [75, 76] est une technique qui permet de remédier au problème de surapprentissage. Cette technique est appliquée au niveau des couches cachées et est utilisée par notre réseau durant la phase d'apprentissage.

Elle consiste à désactiver aléatoirement la sortie d'un ensemble de neurones de la couche cachée où elle est appliquée en mettant leur valeur à zéro par une probabilité définie par l'utilisateur [48, 75]. Par exemple, un « *Dropout* » qui a une probabilité de 0.5 appliqué à une couche cachée désactivera 50% des neurones de cette couche durant la phase d'apprentissage. En procédant ainsi, on arrive à empêcher ou réduire la « *Co-adaptation* » des activations des neurones qui se produit lorsque deux ou plusieurs neurones dépendent les uns des autres durant la phase d'apprentissage les rendant fortement corrélés et rendant notre réseau trop adapté à nos données d'entraînement [4, 75]. En effet, un réseau trop adapté aux données d'apprentissage devient moins performant sur les données de validation qui déterminent sa capacité de généralisation. En clair, l'usage de cette technique donne à notre modèle la capacité de classifier correctement une image, même si certains neurones sont désactivés, [75] et tend à accélérer son apprentissage [48] et à améliorer sa capacité de généralisation.

3.5.5 - Augmentation des données

L'augmentation des données est une technique qui consiste à agrandir artificiellement le nombre de données en créant de nouvelles données [49, 54, 77]. Cela se fait en appliquant différents types de transformations aux données originales. Ces transformations comprennent une série d'opérations dans le domaine de la manipulation d'images telles que le zoom, le pivotement, le redimensionnement et bien d'autres transformations.

Cette technique est utilisée pour réduire le risque de surapprentissage sur des données d'image; elle permet également d'améliorer les performances de notre modèle [4, 5] et d'avoir une bonne capacité de généralisation. En effet, Dominic Masters et Carlo Lushi [65] ont réussi à améliorer les résultats du ResNet-32 sur les données CIFAR-10

[31] en utilisant cette technique et plusieurs autres recherches ont démontré l'efficacité de cette technique [4, 5, 15, 78, 79].

Les images des pages de livres que nous allons recueillir représentent nos données sur lesquelles nous allons appliquer cette technique. Les détails sur nos données seront présentés à la section 4.4 et les transformations que nous avons appliquées sur celles-ci à la section 4.5.3.

CHAPITRE 4. MÉTHODOLOGIE DE LA RECHERCHE

4.1 - Introduction

Dans cette partie, nous allons premièrement décrire les outils que nous avons utilisés pour concevoir notre outil (application web) et visualiser notre base de données; ensuite nous aborderons les démarches entreprises pour recueillir les données et pour les organiser. Puis nous discuterons de la construction de notre réseau de neurones à convolution (VGG16 [3]) et enfin, des différentes configurations que nous avons apportées à notre modèle dans le but d'obtenir des résultats satisfaisants.

4.2 - Framework et Environnement de développement

Étant donné que nous devons concevoir une application web, nous avons sélectionné un EDI (Environnement de Développement Intégré) et des outils côté client (front-end) et côté serveur (back-end) à cette fin.

4.2.1 - EDI (Environnement de Développement Intégré)

Le choix de notre EDI s'est naturellement opéré, étant donné que nous développons une application web au moyen d'un cadriciel (Framework) JavaScript côté client (Angular) et côté serveur (Node.js).

Webstorm [80] est l'EDI que nous avons utilisé; il est fréquemment utilisé pour les langages web tels que HTML, CSS et JavaScript. Il est développé par l'entreprise JetBrains et basé sur la plateforme IntelliJ IDEA. Il offre de nombreuses fonctionnalités facilitant le développement en JavaScript en HTML et CSS.

4.2.2 - Angular et Angular Material

AngularTS [81] est un cadriciel côté client open source basé sur Typescript [82] qui permet de créer des applications à page unique ou SPA (Single Page Applications). Ce cadriciel est codirigé par l'équipe du projet Angular chez Google et est une réécriture complète du langage AngularJS.

Angular Material [83] est un module d'Angular qui permet la création et la personnalisation d'une interface utilisateur flexible et réactive en mettant à la disponibilité du développeur un vaste choix de composants.

4.2.3 - Node.js, Express et Mongoose

Node.js [84] est un langage de bas niveau qui permet l'exécution JavaScript côté serveur. Il utilise le moteur V8 de Google Chrome qui compile (Just In Time) et transforme le code JavaScript très rapidement en code machine et est orienté vers les applications évènementielles. Sa nature non-bloquante lui permet d'effectuer différentes actions et opérations simultanément (n'attend pas qu'une opération soit terminée avant d'en effectuer une autre) Exemple : attente de la réponse d'une requête envoyée à un API.

Express [85] est un micro-Framework qui fournit des outils facilitant la création d'applications Node.js. Il est basé sur le concept des middlewares (qui fournissent des micro-fonctionnalités) et permet par exemple de gérer plus facilement les Routes (URL) et d'utiliser des modèles (*Template*) tels que ejs [86].

Mongoose [87] est un module Node.js qui utilise des *« Schéma »* pour modéliser les données. Il permet de définir les types de variables et de structurer les données; ce qui n'est normalement pas possible avec les systèmes de gestion de base de données non-relationnelles [88].

4.2.4 - Pdf-image

Pdf-image [89] est une librairie node.js qui permet l'extraction d'image (chaque page) au format .png à partir d'un document PDF.

4.2.5 - Tesseract.js

Tesseract.js [90] est une librairie JavaScript qui permet d'extraire le contenu textuel qui se trouve dans une image. Il supporte plus de 100 langues dont le français et l'anglais et peut s'exécuter dans un navigateur et sur un serveur avec Node.js.

4.3 - Base de données

Nous stockons toutes les informations des documents PDF et les images de chacune des pages scannées de nos documents dans une base de données NoSQL appelée MongoDB [91].

4.3.1 - MongoDB Atlas Cloud

MongoDB [91] est un système de gestion de base de données NoSQL qui stocke des documents de type JSON dans des collections (aussi appelé table en SQL). Avec MongoDB les champs peuvent varier d'un document à un autre et la structure des données peut être modifiée. On parle de base dé-normalisée, ce qui est en opposition avec les systèmes de gestion de base de données relationnelles classiques.

MongoDB Atlas [91] est une base de données MongoDB sous forme de service cloud qui permet au développeur de se concentrer sur la création des applications sans tenir compte de certaines tâches opérationnelles liées à la gestion de la base de données que sont la configuration, l'allocation des ressources, les mises à jour, les restaurations et les sauvegardes en cas de problème.

4.3.2 - MongoDB Compass

MongoDB Compass [91] est un GUI (*Graphical User Interface*) qui permet de visualiser et manipuler les données dans la base de données MongoDB Atlas. À partir de ce GUI, nous aurons la possibilité d'ajouter, de modifier, de mettre à jour les documents qui y sont stockés. Il embarque également une panoplie d'autres fonctionnalités telles que le débogage et l'optimisation des performances.

4.3.3 - Collections

Nous allons présenter les différentes collections (équivalent des tables dans les bases de données relationnelles) que nous avons créées dans le cadre de ce mémoire dans le but de stocker les données relatives à nos documents PDF et à notre application.

Pour atteindre l'objectif que nous nous sommes fixés, nous avons créé une base de données nommée **BookDB** qui comporte deux collections :

- La collection ***pdfs*,** stocke les données de chaque document PDF;
- La collection ***pages*,** liée à la première stockera les informations de chaque page de ce document PDF.

4.3.3.1 - Collection *pdfs*

La collection ***pdfs*** comporte 4 champs avec des schémas définis grâce à la librairie Mongoose Node.js :

- **_id** : Ce champ correspond à l'identifiant de chaque document PDF chargé et est généré automatiquement, il s'agit d'un uuid (*Universally Unique Identifiers*). C'est une chaine de caractères unique générée de manière aléatoire nous permettant d'identifier un enregistrement dans une collection.
- **title** : Représente le nom d'un document entré par l'utilisateur après avoir chargé le document PDF depuis l'interface de l'application;
- **documentPath** : Correspond au chemin où notre document est stocké dans le serveur Node.js;
- **totalPages** : Constitue le nombre total de pages de notre document PDF.

Champ	**_id**	**title**	**documentPath**	**totalPages**
Type	String	String	String	Number
Description	Identifiant du document PDF	Nom du document	Chemin d'accès au document	Nombre total de Pages

Tableau 1 – Collection *pdfs*.

4.3.3.2 - Collection *pages*

Notre collection ***pages*** comporte 6 champs avec des schémas définis grâce à notre librairie Mongoose Node.js qui sont les suivant :

- **_id** : Ce champ correspond à l'identifiant de la page du document PDF chargé et est généré automatiquement (uuid);
- **noPage** : Correspond au numéro de la page (commençant par 0);
- **pageContent** : Contenu textuel de la page;
- **imagePath** : Chemin où l'image de la page est stockée;
- **prediction :** Résultat de la prédiction : 0 ou 1 (où 0 correspond à une page contenant des annotations et 1 une page ne contenant pas d'annotation);
- **pdfRef :** Identifiant du document PDF auquel cette page est liée.

Champ	_id	noPage	pageContent	imagePath	prediction	pdfRef
Type	String	Number	String	String	Number	String
Description	Identifiant de la page (uuid)	Numéro de la page	Contenu textuel de la page	Chemin d'accès de l'image	Résultat de la prédiction	Identifiant du document PDF

Tableau 2 - Collection *pages*.

4.4 - Données

Dans cette section, nous allons présenter nos données, la procédure de collecte ainsi que leur organisation.

4.4.1 - Collecte des données

Nous disposons au total de 438 images pour l'entraînement et la validation de nos modèles dont 390 (195 images pour la classe *Annotée* et 195 images pour la classe *Non-annotée*) pour l'entraînement et 48 images (24 pour chacune des classes) pour la validation de nos différents modèles.

Nos données sont équilibrées pour chacune des classes car, comme l'atteste [37] la distribution des données a un grand impact sur les performances et des données équilibrées donnent des résultats optimaux. Concernant nos jeux de données, nous les avons collectés de diverses sources :

- 49 images annotées de [1];
- 49 images non annotées de [92];
- Nous avons extrait 340 images (170 pages annotées et 170 pages non-annotées) manuellement, elles-mêmes tirées de 5 ouvrages du catalogue des livres anciens de l'Université du Québec à Trois-Rivières. Ces 5 ouvrages sont les suivants :
    - Horace, *Q. Horatii Flacci Carmina, ab omni obscoenitate expurgata, cum annotationibus* [*Poésies d'Horace, expurgées de toute obscénité, avec des annotations*], Paris, Brocas et Aumont, 1750;
    - Jean Scapula, *Lexicon graecolatinum novum* [*Nouveau lexique gréco-latin*], Basileae [Bâle], Hervagiana, 1580;
    - Dominique Bouhours, *Les entretiens d'Aristide et d'Eugene*, Paris, S. Mabre-Cramoisy, 1671;
    - Madame de Genlis, *De l'influence des femmes sur la littérature Française, comme protectrices des lettres et comme auteurs, ou Précis de l'histoire des femmes françaises les plus célèbres*, Paris, Maradan, 1811;
    - Bernardin de Saint-Pierre, *Paolo e Virginia*, Firenze [Florence], Molini, 1795.

4.4.2 - Organisations des données

Nos données ont été organisées en deux répertoires principaux (voir Figure 13) qui correspondent à nos deux classes (*Annotée* et *Non-annotée*). Chaque dossier contenant deux sous-dossiers : l'un pour l'entraînement appelé **"training_set"** et l'autre pour la validation appelé **"test_set".**

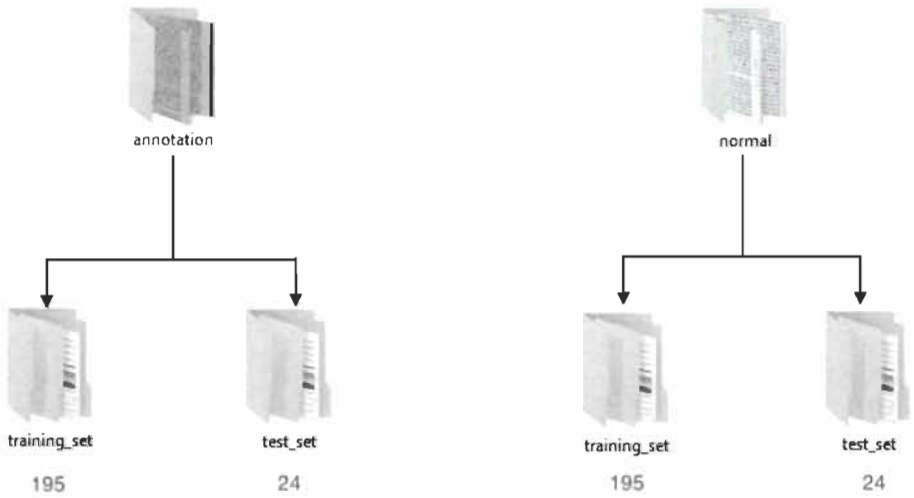

Figure 13 - Organisation des classes de données.

4.4.3 - Données pour le test de l'application

Pour le test de notre application, nous avons recueilli 60 images de pages de livres dont 30 comportant des annotations et 30 n'en comportant pas.

Les images qui comportent des annotations ont été recueillies de la base de données Diva-hisDB [93]. Quant à celles qui ne comportent pas d'annotations, elles ont été recueillies à partir de deux sources [94, 95]. Ces images ont été combinées pour créer un document PDF étant donné que notre application accepte seulement des livres dans ce format. Le nom que nous assignerons à ce document pour le test est « **test data** ».

4.5 - Modèle de classificateurs

Dans la présente section, nous décrirons les outils que nous avons utilisés pour la construction et le déploiement de notre modèle. Nous survolerons également les différentes configurations que nous avons apportées à nos différents modèles dans le but d'améliorer nos résultats et pour obtenir de meilleures prédictions.

4.5.1 - Outils

4.5.1.1 - Anaconda

Anaconda [96] est une distribution Python [97] libre qui intègre un gestionnaire (conda [98]) de paquets permettant d'installer facilement des paquets. Il embarque de nombreuses applications, dont Jupyter notebook [99], essentielles pour la création et la configuration de nos modèles, tout comme pour l'enregistrement et la visualisation des résultats graphiques et expérimentaux des modèles.

4.5.1.2 - Jupyter Notebook

Jupyter Notebook [99] est une application Open Source client-serveur créée par l'organisation Project Jupyter [99] et utilisée pour la création de documents dans plus de 40 langages de programmations dont Python [97].

Les documents sont constitués d'une liste ordonnée de cellules d'entrées, chacune pouvant contenir du texte au format Markdown [100], des formules mathématiques et du code informatique exécutable.

Nous allons l'utiliser pour la création et l'ajustement de notre modèle de classificateur parce qu'elle dispose d'outils permettant de projeter les résultats sous forme de graphiques et cartographies interactifs et de tableaux et aussi d'enregistrer des documents d'analyse.

4.5.2 - Bibliothèques

Les principales bibliothèques que nous avons utilisées pour la construction de nos modèles sont les suivantes :

Keras [101] est une bibliothèque open source écrite en Python [97] servant d'interface avec d'autres librairies d'apprentissage machine que sont : TensorFlow, CNTK [102] et Theano [103]. Elle permet l'interaction et le prototypage facile et rapide des algorithmes d'apprentissage automatique et profonds tel que le réseau de neurones convolutif VGG16

[3] que nous avons utilisé dans ce mémoire. Elle fonctionne également de façon transparente sur CPU et GPU.

Tensorflow [104] est une bibliothèque open-source développée par l'équipe Google Brain qui regroupe un grand nombre de modèles et d'algorithmes d'apprentissage machine et profond. Tensorflow en version (2.0.0) est l'interface que nous avons utilisée avec Keras.

Matplotlib [105] est une bibliothèque Python [97] qui nous permettra de tracer et de visualiser les résultats de l'entraînement et de la validation de nos modèles de manière graphique.

4.5.3 - Prétraitement de données

La taille de toutes nos images sera réduite au format (224x224) grâce au module **"ImageDataAugmentation"** de Keras à l'aide de la méthode **"flow_from_directory()"**. Nous procédons ainsi car notre modèle VGG16 [3] prend des images de tailles fixes en entrée, comme nous l'avons mentionné dans la section 3.4.

Nous avons utilisé le même module et la même méthode pour spécifier la taille du lot et pour appliquer l'augmentation des données sur nos données d'apprentissage et de validation.

L'augmentation des données est la technique que nous avons abordée à la section 3.5.5 qui permet de compenser la quantité de données limitées [1] étant donné que nous n'avons que 390 images pour l'entraînement et 48 images pour la validation de notre modèle.

Les transformations que nous avons appliquées à nos données d'entraînement sont les suivantes : **rescale, shear_range, zoom_range et horizontal_flip**. Seul le **rescale** a été appliqué sur nos données de validations. La figure 14 nous donne un aperçu de l'application de l'augmentation des données.

Original

Après application de image augmentation

Figure 14 - Exemple d'application de l'augmentation d'image.

4.5.4 - Configurations et architectures des modèles

Nous avons procédé à 8 expérimentations dans le but d'obtenir un modèle à la fois stable et performant, c'est-à-dire qui nous donne de bons résultats et qui a une bonne capacité de généralisation. Nos modèles ont été entrainés sur le système d'exploitation Windows 10 avec un GPU NVIDIA Geforce GTX 1080 Ti.

Pour toutes nos expérimentations, nous avons appliqué l'augmentation d'image que nous avons décrite à la section 3.5.5 et 4.5.3 et nous avons réduit la taille de toutes nos images pour l'apprentissage et la validation de notre modèle.

Nous changerons le nombre de nœuds de la dernière couche de notre VGG16 [3] modèle qui comporte 1000 nœuds, étant donné que ce modèle a été conçu originellement pour classifier la base de données ImageNet [9] en 1000 classes; la dernière couche comportera donc 1 nœud, puisque nous rappelons que nos images sont classifiées en deux classes (Annotée : 0 ou Non-annotée : 1).

Les poids associés à cette base de données seront chargés pour toutes nos expérimentations. La fonction d'activation de nos couches cachées est la fonction *Relu* et pour la dernière couche (couche de sortie), sigmoïde.

L'algorithme d'optimisation est la descente du gradient stochastique (SGD) avec un moment de 0.9, tout comme l'article [7] et la fonction de perte utilisée est le *« binary_cross_entropy ».*

4.5.4.1 - 1re expérimentation

Pour notre première expérimentation, nous avons ré-entrainé seulement la dernière couche de notre modèle et avons utilisé une taille de lot égale à 32 et un taux d'apprentissage de 0.001. Notre modèle est entrainé sur 250 itérations *« epochs ».* Nous obtenons ainsi 4097 paramètres à entraîner sur 134.264.641 paramètres au total.

4.5.4.2 - 2e expérimentation

Notre seconde expérimentation est légèrement différente de la première. Nous avons utilisé la même taille de lot et nous l'avons entrainé pour le même nombre d'itérations *« epochs ».* Le taux d'apprentissage a été réduit à 0.0001 et nous avons entrainé les 3 dernières couches au lieu de la dernière couche. Ainsi 119.549.953 paramètres seront entrainés sur un total de 134.264.641 paramètres.

4.5.4.3 - 3e expérimentation

Pour cette expérimentation, nous avons ajouté un paramètre qui est le *« Dropout »* qui a une valeur de 0.5 entre la première couche de réseau de neurones artificiels et la seconde couche. Tous les autres paramètres restent les mêmes que lors de la seconde expérimentation.

4.5.4.4 - 4e expérimentation

Notre modèle pour cette expérimentation a les mêmes paramètres que la précédente, mais nous avons ajouté un *« Dropout »* supplémentaire de la même valeur 0.5 entre la 2e couche et la dernière couche du réseau de neurones artificiel et nous l'entrainons pour 300 itérations *« epochs »* au lieu de 250, comme c'est le cas pour les 3 premières expérimentations.

4.5.4.5 - 5e, 6e et 7e expérimentation

Pour la 5e, 6e et 7e expérimentation, nous entrainons nos modèles avec des tailles de lot de 16, 8 et 4 successivement. Deux *« Dropout »* sont utilisés tout comme dans l'expérimentation 4 et les modèles sont entrainés pour le même nombre d'itérations *« epochs »* et avec le même taux d'apprentissage.

4.5.4.6 - 8e expérimentation

Pour cette expérimentation, des lots de différentes tailles sont utilisés (32 pour les données d'entraînement et 8 pour celles de validation). Quant aux autres paramètres, ils demeurent les mêmes que lors des expérimentations 5, 6 et 7.

4.5.5 - Mesures d'évaluation

Pour l'évaluation de nos différents modèles, nous avons utilisé les mesures de performance suivantes :

- Accuracy (l'exactitude) : correspond aux performances de notre modèle sur l'échantillon d'entraînement;

- Val_accuracy : correspond aux performances du modèle sur l'échantillon de validation; il donne un aperçu des capacités de généralisation du modèle.

Pour notre problème de classification binaire, le taux d'exactitude sur nos échantillons d'entraînement et de validation est déterminé par le calcul suivant :

$$\frac{VP + VN}{VP + FN + VN + FP}$$

Où VP = Vrais positifs, VN = Vrais négatifs, FP = Faux positifs, et FN = Faux négatifs.

VP (Vrais positifs) : La prédiction est positive, et la valeur réelle est positive.

VN (Vrais négatifs) : La prédiction est négative, et la valeur réelle est négative.

FP (Faux positifs) : La prédiction est positive, mais la valeur réelle est négative.

FN (Faux négatifs) : La prédiction est négative, mais la valeur réelle est positive.

4.5.6 - Flask API

Flask API [106] est un microservice qui permet un développement web plus facile en Python [97]. Notre modèle étant construit à l'aide du langage Python, des librairies et des technologies que nous avons explorées dans les sections précédentes, nous allons le charger préalablement et le déployer afin qu'il soit prêt à répondre aux requêtes qui lui sont envoyées depuis notre API Express Node.js.

4.6 - Intégration

Lorsqu'un livre PDF est chargé depuis notre front-end Angular, une requête est envoyée à notre API Express Node.js, qui stocke le document PDF et certaines informations liées de ce PDF dans les différents champs de la collection ***pdfs***.

Ensuite, chaque page scannée de ce document ou livre sera extraite à l'aide de la librairie **pdf-image** et son contenu océrisé grâce à la librairie **Tesseract.js.**

Enfin, pour chacune des images de pages du PDF, une requête est envoyée à notre API Python Flask et une réponse correspondant au résultat de la classification (0 : présence d'annotation ou 1 : absence d'annotation) est retournée à notre API Express Node.js, qui stocke les informations liées à chacune des images de pages dans la collection ***pages***, comme l'illustre la figure 15.

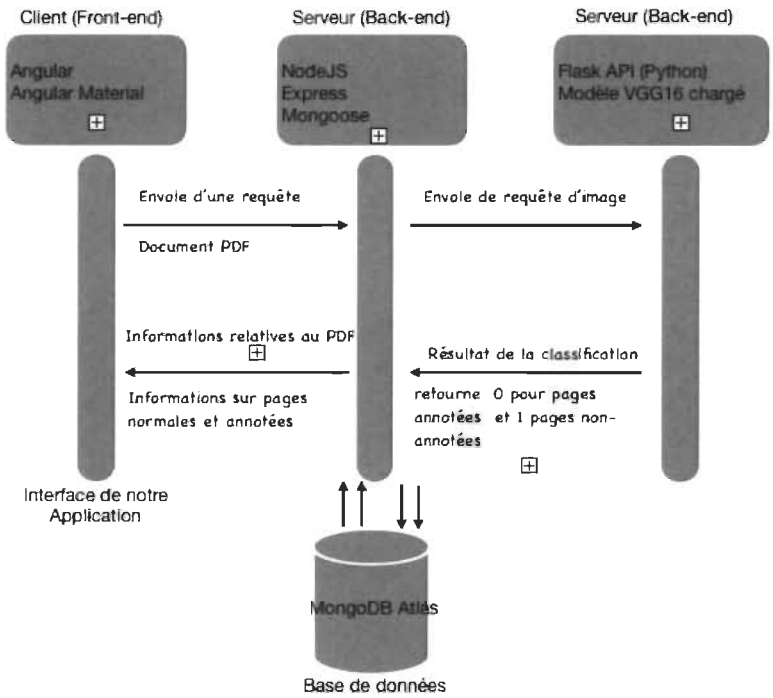

Figure 15 – Flux d'action de l'application.

CHAPITRE 5. RÉSULTATS ET INTERPRÉTATIONS

5.1 - Présentation des résultats et interprétations

Les figures 16 à 23 nous présentent le taux d'exactitude sur nos données d'entraînement (Accuracy) et sur nos données de validation (val_accuracy) pour nos différents modèles (expérimentations). Ce sont les performances sur notre échantillon de validation qui nous intéressent car celles-ci nous donnent un aperçu des capacités de généralisation de notre modèle.

Les trois premières expérimentations : 1, 2 et 3 (figure 16, 17, 18) utilisent la même taille de lot qui est de 32 sur nos données d'apprentissage et de validation et sont entrainées pour 250 itérations *« epochs »*. Nous observons que la réduction du taux d'apprentissage à la 2^{e} expérimentation (figure 17) et le réentraînement du modèle sur nos 3 dernières couches au lieu de la dernière couche, comme c'est le cas pour la 1re expérimentation (figure 16), permet d'améliorer la stabilité du réseau sur nos données d'apprentissage et ceux de validation, mais aussi d'améliorer les performances de notre réseau sur ces deux ensembles.

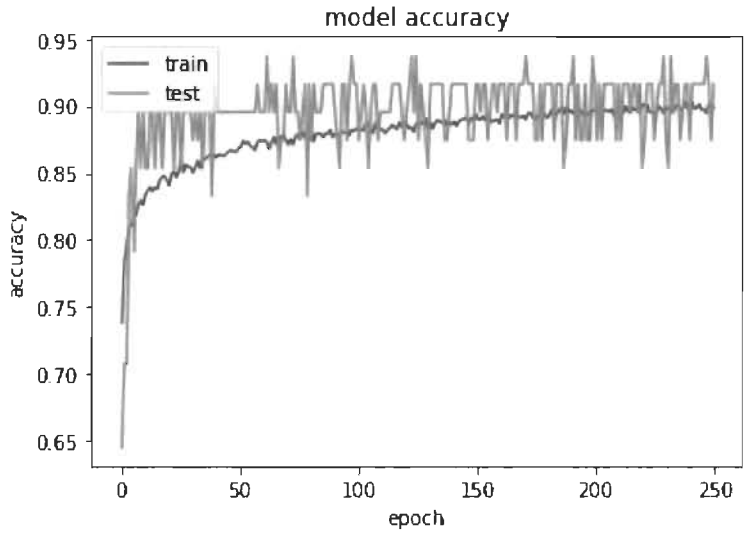

Figure 16 - Résultats de la première expérimentation.

Le taux d'exactitude sur nos données d'entraînement qui peinait à atteindre 90% a continué à progresser jusqu'à atteindre 99.97% en fin d'entraînement pour notre 2e expérimentation (figure 17). Les oscillations du taux d'exactitude de nos données de validation (val_accuracy) dans l'expérimentation 1 (figure 16) se sont significativement réduites lors de la 2e expérimentation (figure 17). Le taux d'exactitude sur les données de validation de la 2e expérimentation (figure 17) présente de nombreuses zones de convergence. Après seulement 50 itérations « *epochs* », le modèle commence à converger sur nos données de validation et on observe des performances stables de 93.75% sur cet ensemble pendant 22 itérations (précisément entre la 53e et la 74e).

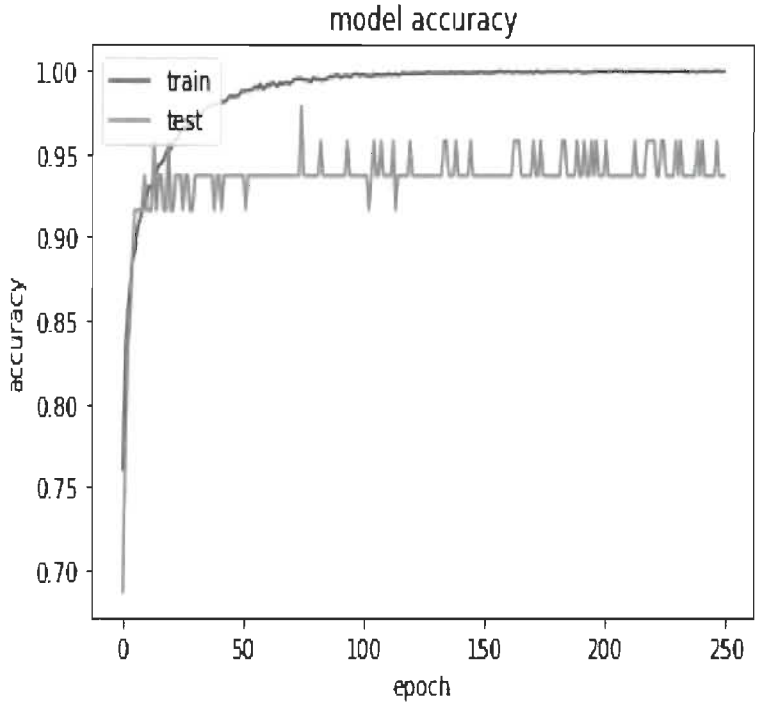

Figure 17 - Résultats de la 2e expérimentation.

L'ajout d'un *« Dropout »* entre les couches cachées à la 3e expérimentation (figure 18) et de deux *« Dropout »* à la 4e expérimentation (figure 19) augmentent les zones de convergence de nos modèles sur nos données de validation. Le taux d'exactitude sur nos données d'entraînement quant à lui pour nos deux expérimentations est moins lisse et légèrement dégradé par rapport à la 2e expérimentation (figure 17) lorsque nous ajoutons le *« Dropout »*. On peut donc conclure que l'ajout du *« Dropout »* a une incidence négative légère sur les performances de nos données d'entraînement, mais améliore la convergence et les capacités de généralisation de nos modèles.

L'entraînement pour l'expérimentation 2 (figure 17), l'expérimentation 3 (figure 18), l'expérimentation 4 (figure 19) pourrait être interrompu à l'aide de méthodes telles que le *« Early stopping »* [107], lorsque nos modèles semblent convergés tel qu'énoncé à la section 3.2.5.

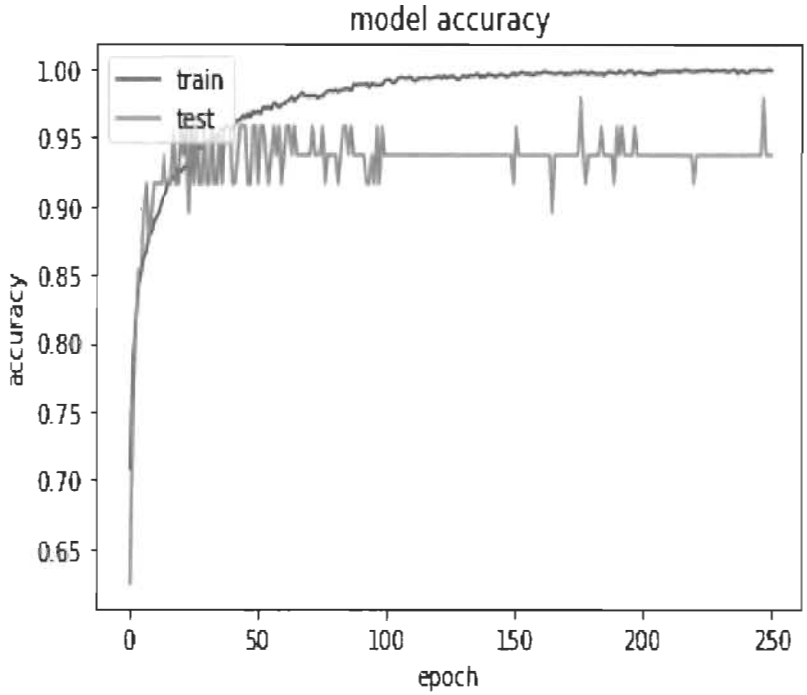

Figure 18 - Résultats de la 3e expérimentation

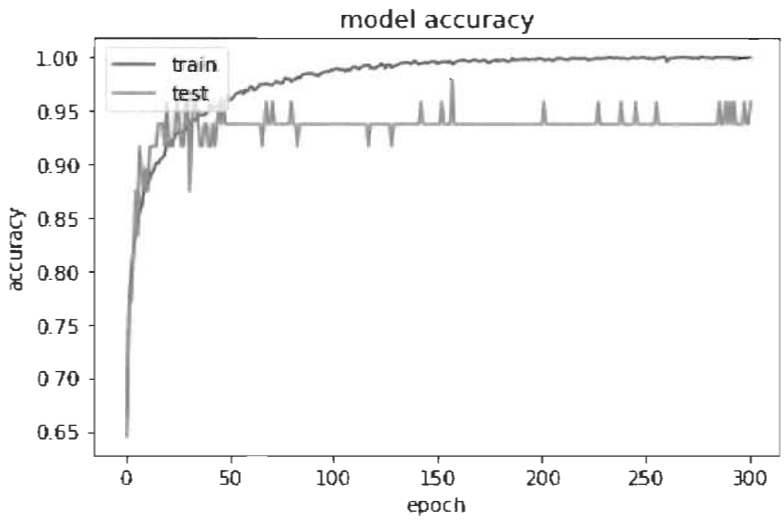

Figure 19 - Résultats de la quatrième expérimentation.

Pour les expérimentations 5 (figure 20), 6 (figure 21) et 7 (figure 22), nous assignons respectivement les tailles de lot de 16, 8 et 4; concernant les autres configurations, elles restent les mêmes que lors de la 4e expérimentation (figure 19). Nous observons pour ces 3 expérimentations que plus nous réduisons la taille de lot, plus les performances sur nos deux ensembles de données (accuracy et val_accuracy) se dégradent.

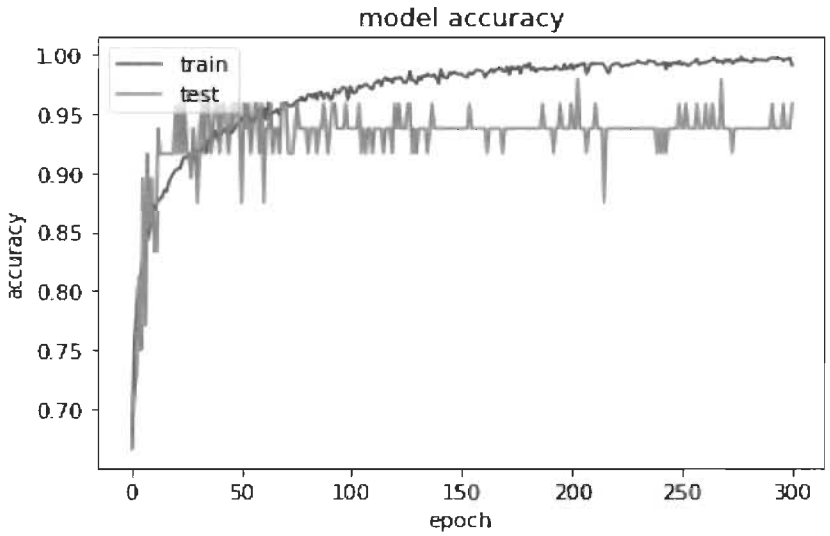

Figure 20 - Résultats de la 5e expérimentation.

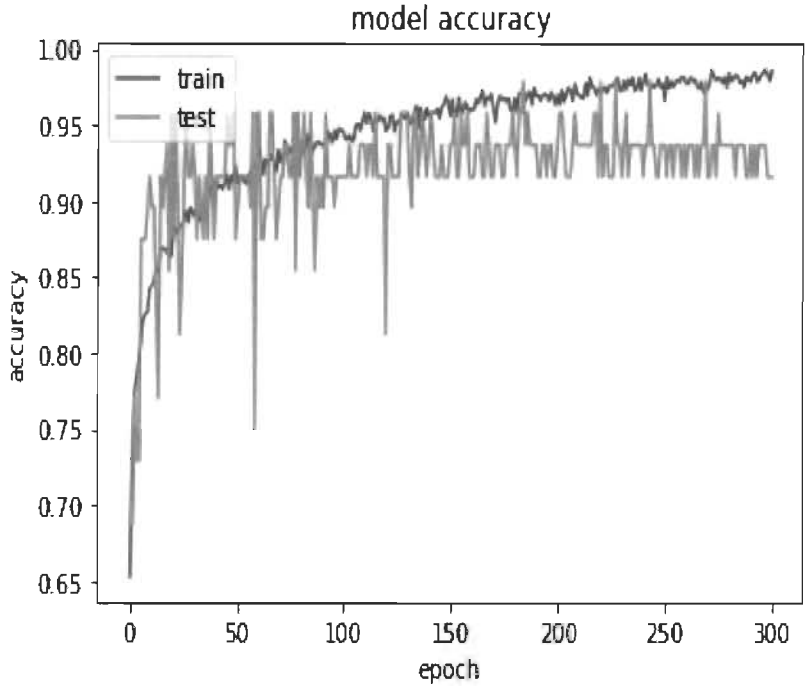

Figure 21 - Résultats de la 6e expérimentation.

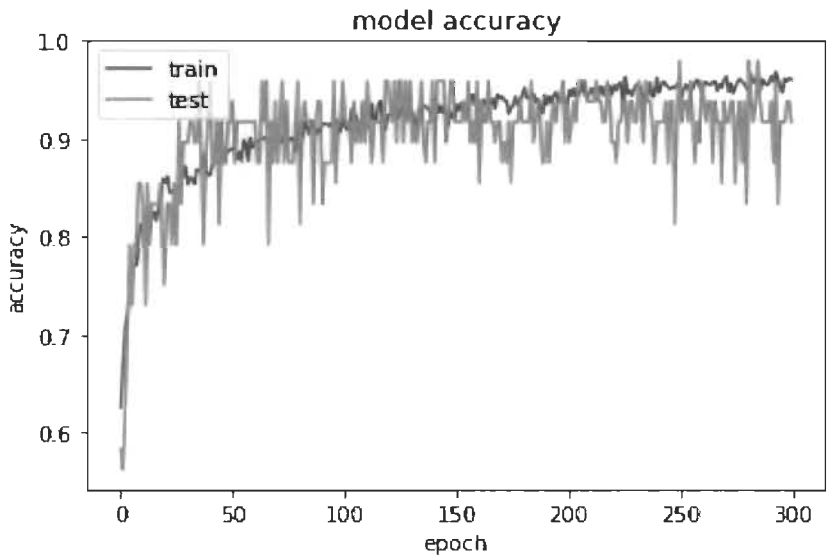

Figure 22 - Résultats de la 7e expérimentation.

Nous utilisons une taille de lot de 32 sur nos données d'entraînement et de 8 sur notre échantillon de validation pour notre dernière expérimentation (figure 23).

Nous observons que notre réseau a une convergence lente sur les données de validation mais plus longue comparé aux expérimentations 3 (figure 18) et 4 (figure 19); en effet, les performances de ce modèle sur nos données de validation demeurent stables pendant plus de 120 itérations. Nous obtenons ainsi un taux d'exactitude de 93.75% sur nos données de validation et de 99.96% sur nos données d'entraînement en fin d'entraînement.

Ces résultats confèrent à notre modèle une excellente capacité de généralisation. Nous allons le déployer et l'intégrer à l'API Flask de notre application pour effectuer le test sur des données qui n'ont pas encore été vues par le classificateur obtenu.

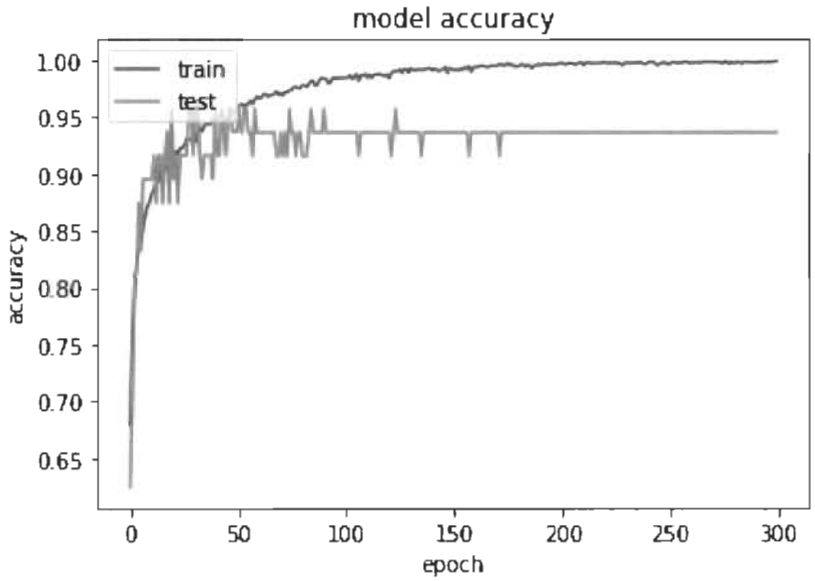

Figure 23 - Résultats de la 8e expérimentation.

5.2 - Outils d'aide à la numérisation

5.2.1 - Interface finale de l'application

Notre outil est une application web composée de trois interfaces principales :

La première interface est celle qui nous permet d'importer un nouveau document PDF dans notre base de données. Elle est constituée d'un formulaire (voir

figure 24) qui donne la possibilité à l'utilisateur de donner un nom au document PDF. Ce formulaire contient également deux boutons permettant de (1) sélectionner un document PDF et (2) d'effectuer l'enregistrement du document.

Lorsque l'utilisateur tente de sélectionner un document qui est dans un format différent, un message d'erreur lui est adressé lui demandant de sélectionner un document au format approprié.

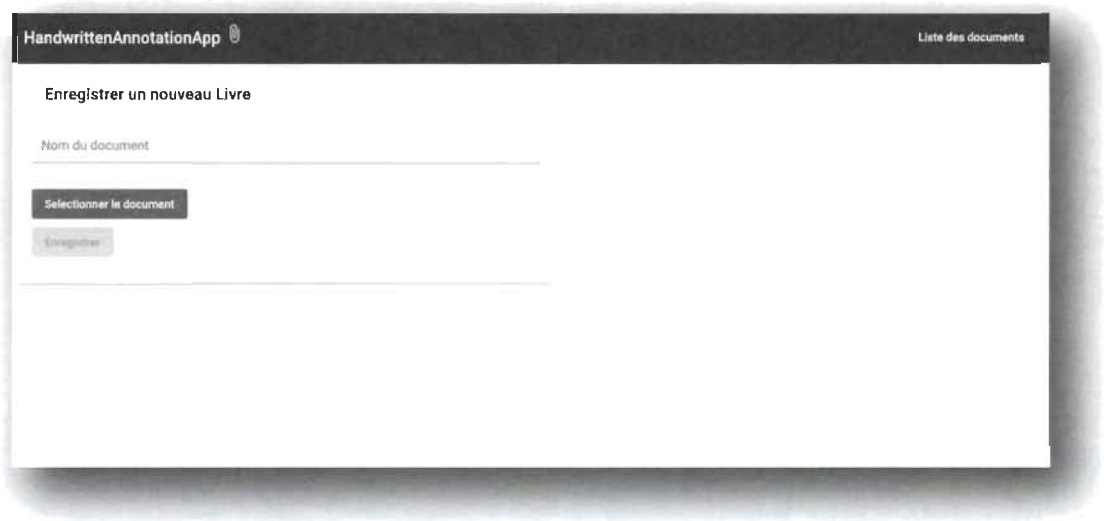

Figure 24 - Interface d'importation de document de l'application.

Une fois que le document est enregistré avec succès, l'utilisateur est dirigé vers la 2e interface (voir figure 25). Cette dernière permet la visualisation des différents documents que nous avons enregistrés depuis notre application. La figure 25, nous présente 3 documents qui ont été enregistrés.

Figure 25 - Interface d'affichage de la liste des documents.

Pour chacun des documents, l'utilisateur a la possibilité de connaitre le nombre total de pages, de supprimer le document et d'ouvrir le document en sélectionnant la page désirée. L'ouverture du **Document 1** (de l'exemple) correspondant à nos figures 25 et 26. Il comporte 3 pages au total. Une liste déroulante permet à l'usager d'accéder aux pages de ce document (voir figure 26).

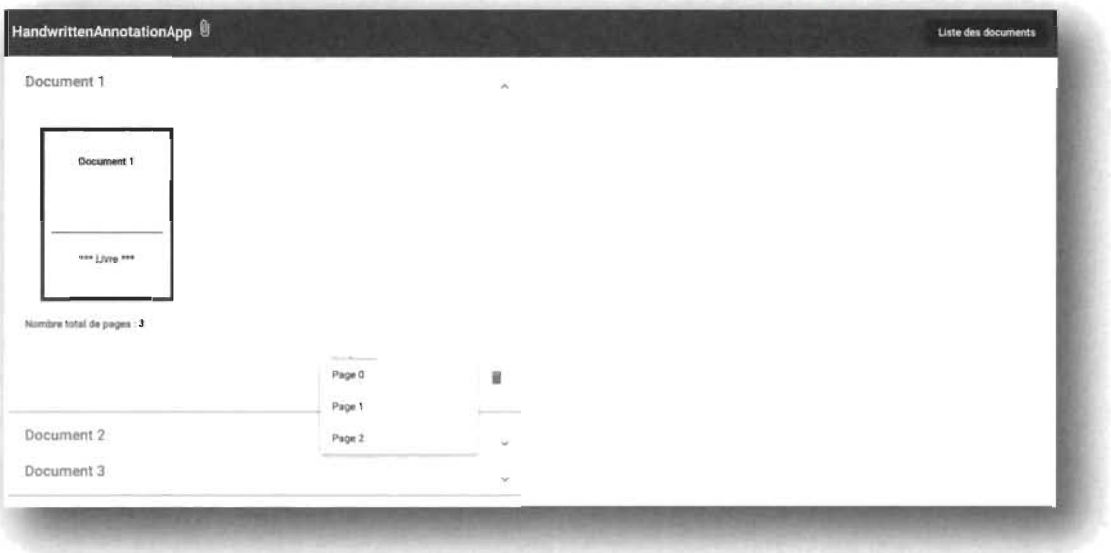

Figure 26 - Un document avec un panel ouvert.

Lorsque l'utilisateur ouvre une page, il est redirigé vers la dernière interface (figure 27). Cette interface permet de naviguer entre les différentes pages de notre document à l'aide de deux boutons précédant et suivant (figure 27). Elle présente à gauche le résultat de la transcription de la page dans un éditeur de texte et à droite l'image de la page scannée en cours du document. Le résultat de la transcription est présenté dans un éditeur de texte pour permettre à l'utilisateur de modifier manuellement le texte extrait et de le mettre à jour grâce au bouton ***update*** qui se trouve en dessous de l'éditeur de texte (figure 27).

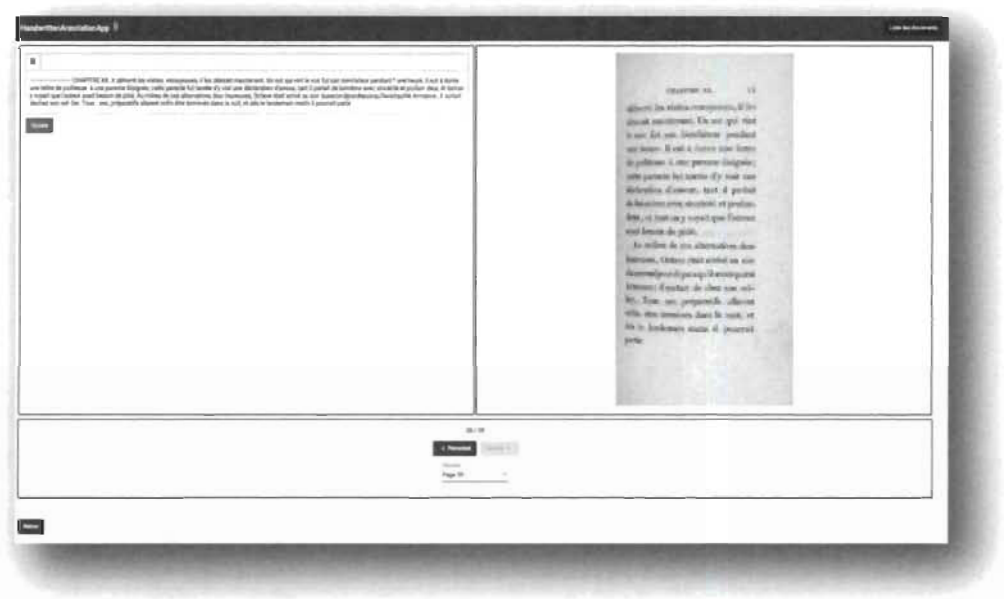

Figure 27 - Interface de navigation entre les pages.

Au bas de cette page (figure 28) se trouve un bouton qui permet de sélectionner la page du document à laquelle l'on veut se rendre, sans avoir à utiliser les deux boutons de navigation. Ce bouton intègre les résultats de la prédiction de nos différentes pages. L'utilisateur peut savoir quelles sont les pages qui sont annotées et celles qui ne le sont pas. Celles qui sont annotées sont surlignées (couleurs **indigo** sur la figure 28).

Figure 28 - bouton affichant les différentes pages.

5.2.2 - Résultats du test de l'application

Nous procédons au test de notre modèle à l'aide d'un document PDF qui est constitué de 60 pages (30 pages pour chacune des catégories) sous le nom « **test data** » comme mentionné à la section 4.4.3. La figure 29 nous montre que le document a bien été enregistré sous le nom que nous lui avons donné dans notre application : on peut observer le nombre de pages de ce document qui est de 60 en ouvrant le panel prévu à cet effet (voir figure 29).

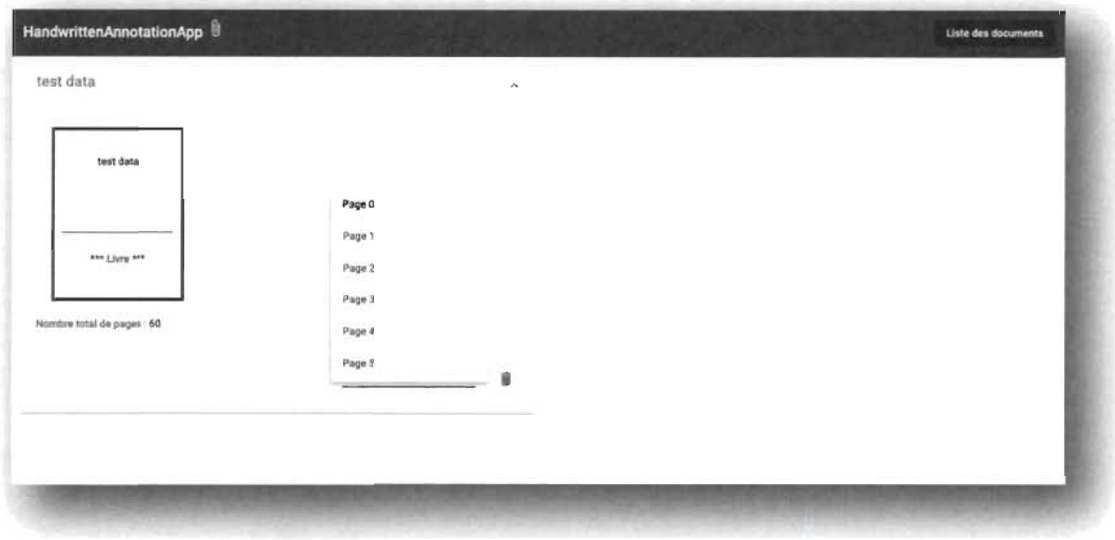

Figure 29 - Document dans la liste des documents.

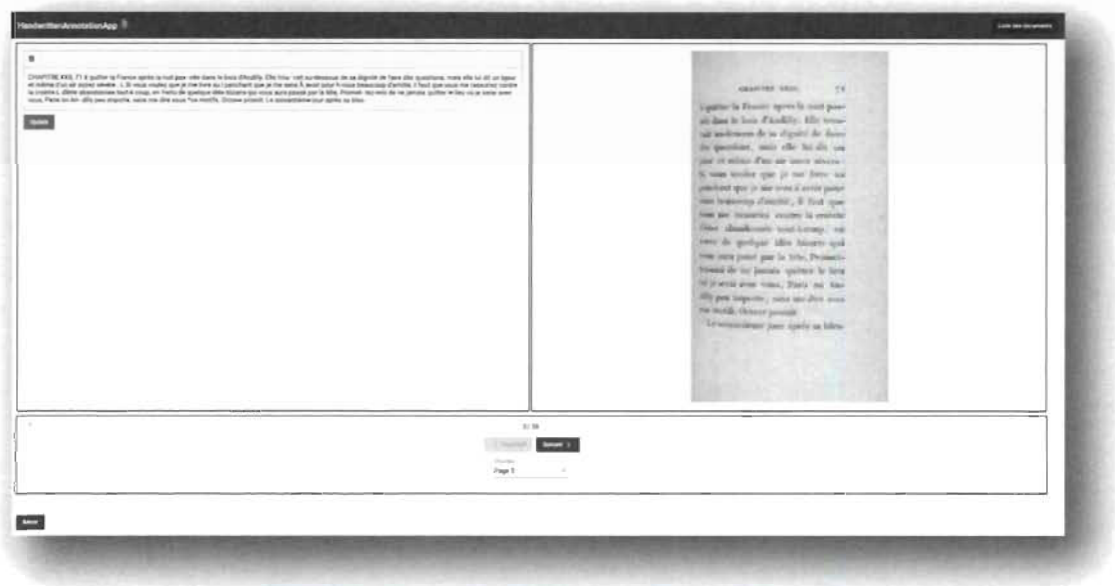

Figure 30 - Visualisation de la première page du document PDF.

La figure 30 nous montre la première page du document et sa transcription textuelle sur l'interface qui nous permet de naviguer entre les différentes pages.

Les figures 31 et 32 nous présentent nos 2 collections (***pdfs*** et ***pages***) sur l'interface du logiciel MongoDB Compass. Notre document et ses différentes pages ont bien été enregistrés dans nos deux collections.

La collection ***pdfs*** (figure 31) comporte les informations liées à notre document telles que l'identifiant **(_id)**, le nom que nous avons donné au document **(title)** : **test data**, le chemin d'accès du document **(documentPath)** et le nombre de pages de ce document **(totalPages)** qui est de **60**.

Quant à la collection ***pages*** (figure 32), toutes les informations liées aux pages du document (voir section 4.3.3) y sont enregistrées. Les champs qui nous intéressent dans cette collection sont :

- **prediction** qui comporte les résultats retournés par notre API Flask;
- **pageContent** qui comporte les résultats de l'extraction textuelle par notre librairie Tesseract.js pour chacune des pages du document.

Le champ **prediction** comporte soit 0 pour la classe **Annotée** soit 1 pour la classe **Non-annotée** pour chacune des images (figure 32).

Figure 31 - Contenu de la collection *pdfs* après le test.

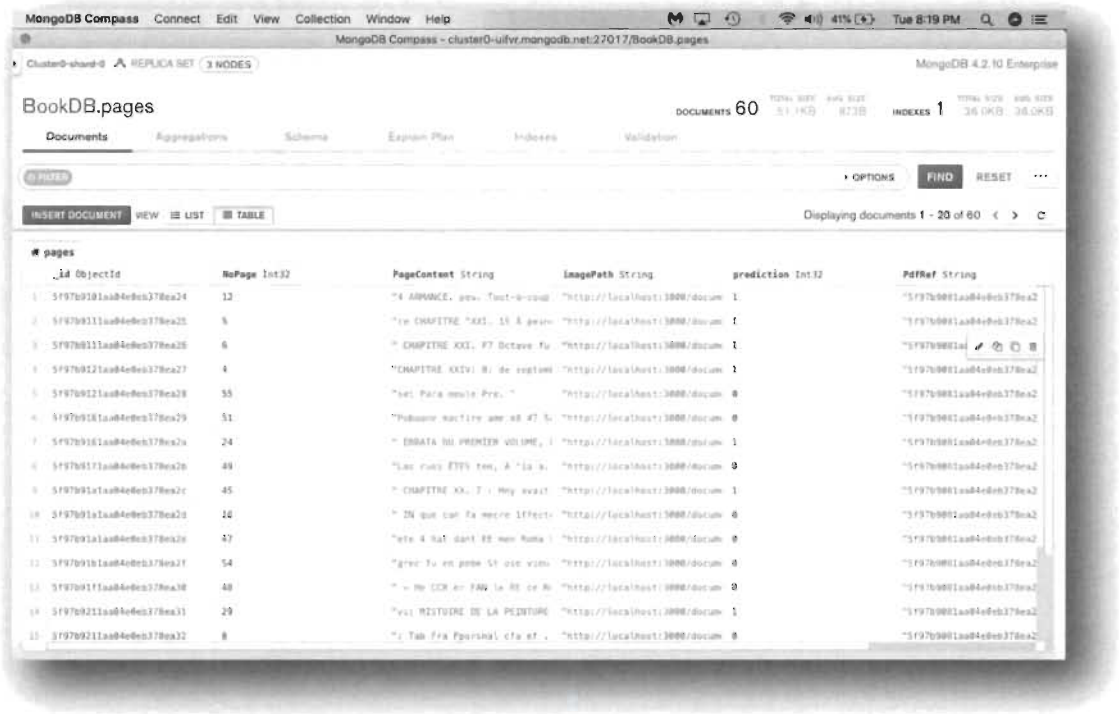

Figure 32 - Contenu de la collection *pages* après le test.

Ces résultats ont été intégrés au bouton (figure 28) sur la page qui nous permet de naviguer entre les différentes pages et de voir les pages annotées. Lorsque nous visualisons notre collection ***pages***, sur MongoDB Compass, on observe pour les pages 7, 8, 9 et 10 par exemple, qu'elles ont été prédites par notre modèle comme appartenant à la classe 0 (Annotée). La figure 33 nous montre qu'elles sont surlignées dans la liste des pages lorsqu'on clique sur le bouton de notre application.

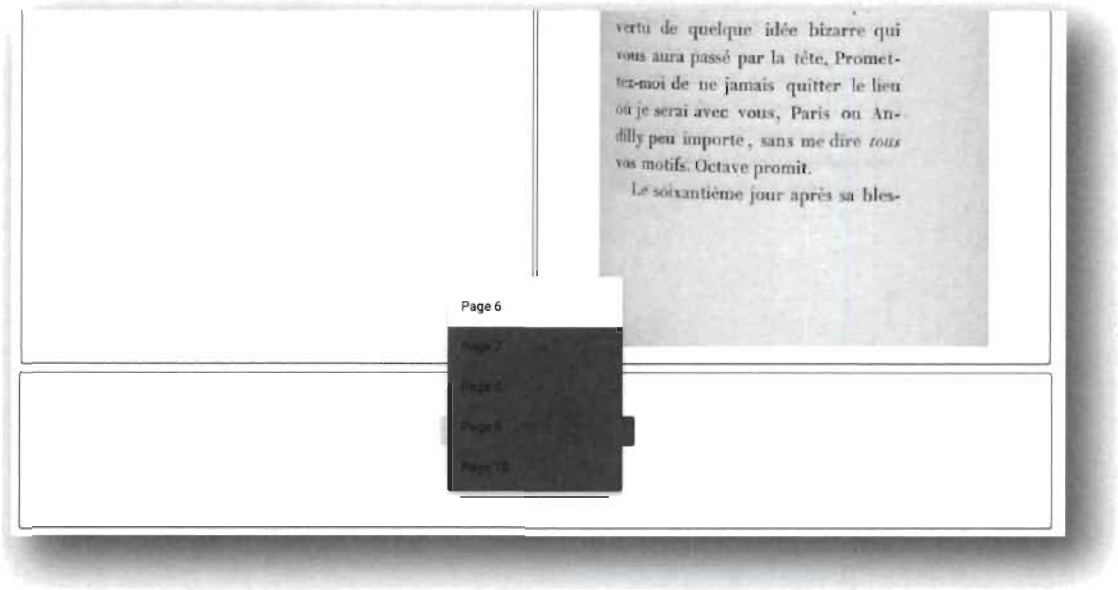

Figure 33 - Liste de quelques pages annotées.

Les résultats de la prédiction de toutes les pages du document peuvent être résumés dans ce tableau :

-	**0**	**1**	**Total**
0	26	0	26
1	4	30	34
Total	30	30	60

Tableau 3 - Résultats de la prédiction dans la matrice de confusion.

Sur nos 30 pages qui comportent des annotations, 26 ont été prédites comme appartenant à la classe 0 (**Annotée**), c'est-à-dire qu'elles ont été correctement prédites par notre modèle, et il y'en a 4 qui ont été prédites comme appartenant à la classe 1 (**Non-annotée).**

Quant aux 30 autres images qui ne comportent pas d'annotations, elles ont toutes été correctement prédites par notre modèle. Étant donné que nos données sont équilibrées, nous utiliserons le taux d'exactitude pour évaluer nos données de test (voir section 4.5.5) :

$$Taux\ d'exactitude = \frac{26+30}{26+4+30+0} = \frac{56}{60} = 93.33\%$$

Nous obtenons ainsi un taux d'exactitude de 93.33% sur nos données de test.

5.2.3 - Résultats de la transcription textuelle

La figure 34 nous montre les résultats de la transcription textuelle de notre première page.

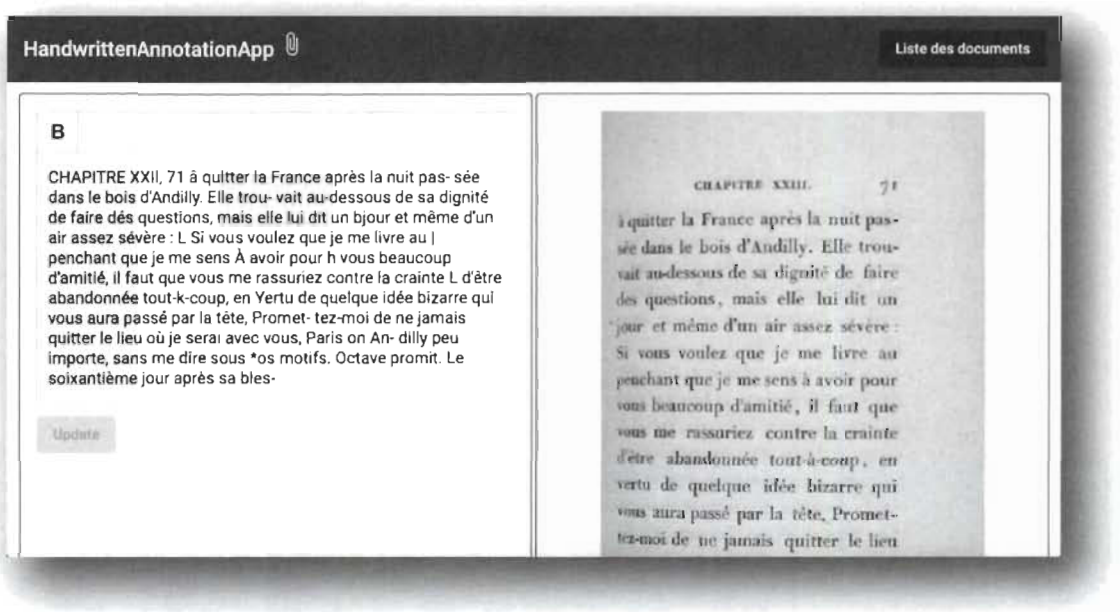

Figure 34 - transcription textuelle de notre première page.

On observe pour cette page que la librairie Tesseract.js intégrée à notre application arrive dans la globalité à reconnaître le texte contenu sur cette page, mais qu'elle commet quelques erreurs et que ces erreurs pour la plupart sont dues au texte qui se trouve dans des zones de la page qui sont légèrement inclinées lors de la numérisation, rendant ainsi la transcription plus difficile.

En observant les résultats de la transcription textuelle de toutes nos pages, on observe que la librairie arrive à détecter les images qui ont un fond plus clair, mais lorsque le fond est dégradé, elle éprouve des difficultés. Elle n'arrive parfois à ne rien détecter lorsque le fond est très dégradé, comme c'est le cas à la figure 35. Les pages des

manuscrits anciens sont généralement de mauvaise qualité et dégradées à cause du temps et de plusieurs autres facteurs.

Des techniques de prétraitement d'images dans certains cas pourraient être utilisées pour améliorer la qualité de certaines images dans le but d'améliorer la transcription du texte.

Figure 35 - Échec de la transcription textuelle.

CHAPITRE 6. DISCUSSIONS ET CONCLUSION

Dans le but d'aider et d'assister les équipes travaillant sur le livre ancien, mais aussi de leur offrir des outils d'investigation performants, nous avons développé un outil qui permet d'analyser des pages de livres anciens scannées, de les retranscrire et de déterminer la présence ou non d'annotations manuscrites. La détection d'annotation est un problème de classification binaire avec les catégories *Annotée et Non-annotée*. Pour classifier automatiquement ces pages, nous avons effectué 8 expérimentations utilisant le VGG16 [3] pré-entrainé sur la base de données ImageNet [9, 25] à l'aide du Transfer Learning. Pour chacune de ces expérimentations, nous avons utilisé des techniques telles que l'augmentation de données et le *« Dropout »* et avons testé différents taux d'apprentissage et tailles de lot. Les performances des modèles ont été évaluées sur un jeu de données qui comprend 438 images de livres anciens pour l'entraînement et la validation de nos modèles dont 390 (195 images pour chacune des classes) pour l'entraînement et 48 images (24 pour chacune des classes) pour la validation de nos différents modèles. Nous avons obtenu un modèle stable qui a une bonne capacité de généralisation avec un taux d'exactitude de 93.75% sur nos données de validation grâce à l'utilisation d'un batch de différentes tailles sur nos données d'entraînement et de validation.

Nous avons démontré ainsi qu'il est possible d'obtenir d'excellentes performances même lorsque le nombre de données est limité et sans prétraitement des données d'entrée. Notre modèle final a été intégré à un moteur de classification sous forme d'API web pour permettre son test concret sur des pages de livres scannées qui n'ont pas encore été vues par le classificateur et pour la transcription textuelle de leurs contenus. Nous obtenons un taux d'exactitude de 93.33% pour la détection de pages annotées à la suite de ce test sur 60 de pages de livres scannés.

Cette application peut être mise à la disposition des chercheurs pour les assister dans le suivi des annotation manuscrites en détectant de manière automatique les pages dans des livres anciens scannées comportant des annotations manuscrites.

Recherches futures

Nous avons introduit dans ce mémoire une nouvelle base de données qui contient 170 pages scannées de livres comportant des annotations manuscrites que nous avons recueillies manuellement dans 5 ouvrages du catalogue des livres anciens de l'Université du Québec à Trois-Rivières. Cette base de données peut être utilisée pour des recherches futures dans le domaine des annotations manuscrites.

Il serait judicieux également d'effectuer des recherches et des expérimentations sur l'usage d'une taille de lot de différentes tailles pour les données d'entraînement et de validation pour confirmer l'efficacité de cette technique car, dans ce mémoire, nous avons effectué un nombre limité d'expérimentations à l'aide de cette technique. Cette technique pourrait, si son efficacité a été prouvée, permettre d'améliorer les résultats de modèles sur différentes tâches, notamment sur la tâche de classification de documents sur la base de données RVL-CDIP [27, 46].

Nous avons utilisé l'algorithme stochastique du gradient et le moment pour toutes nos expérimentations et un taux d'apprentissage fixe de 0.001(10e-3) pour la première expérimentation et de 0.0001(10e-4) pour toutes les 7 autres expérimentations; l'usage de l'optimiseur Adam et d'un taux d'apprentissage graduel (Gradual Learning Rate Decay) pourrait être utilisé sur les données de validation de notre jeu de données tel que proposé par Arindam Das *et al* [8] ou l'usage d'un taux d'apprentissage cyclique proposé par N. Smith [62] pourrait être utilisé dans le but d'améliorer les performances de notre modèle.

Pour accélérer l'entraînement de notre réseau et pour améliorer les capacités de généralisation de notre modèle, l'utilisation du Batch Normalisation pourrait être envisagée. Cette technique est tolérante à l'utilisation des taux d'apprentissage élevés et, dans certains cas, [108] peut éliminer le besoin d'utiliser le « *Dropout* »; elle permet d'améliorer la convergence des « *Batch* » de grandes tailles [109, 74].

Plus de données pourraient être fournies à notre modèle dans le but de le rendre plus robuste et il serait aussi important de tester ses performances sur un nombre conséquent

de données, mais aussi sur d'autres types de documents où la présence ou non d'annotations manuscrites est recherchée. On peut également penser à une solution où l'archiviste peut donner un feed-back sur les prédictions erronées/justes afin d'en tenir compte pour améliorer notre modèle dans le moteur de prédiction de l'API Flask.

CHAPITRE 7. RÉFÉRENCES BIBLIOGRAPHIQUES

[1] A. Kölsch, A. Mishra, S. Varshneya et M. Z. Afzal, «Recognizing Challenging Handwritten Annotations with Fully Convolutional Networks,» *arXiv:1804.00236.*

[2] Y. LeCun, P. Haffner, L. Bottou et Y. Bengio, «Object Recognition with gradient Based learning».

[3] A. Zisserman et K. Simonyan, «Very deep convolutional networks for large-scale image recognition,» *arXiv preprint arXiv:1409.1556, 2014. 1, 3.*

[4] A. Krizhevsky, I. Sutskever et G. E. Hinton, «ImageNet classification with deep convolutional neural networks,» *In Advances in Neural Information Processing Systems 25, NIPS 2012, 2012.*

[5] K. He, X. Zhang, S. Ren et J. Sun, «Deep Residual Learning for Image Recognition,» *arXiv:1512.03385.*

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke et A. Rabinovich, «Going Deeper with Convolutions,» *arXiv:1409.4842.*

[7] M. Z. Afzal et al., «Cutting the Error by Half: Investigation of Very Deep CNN and Advanced Training Strategies for Document Image Classification,» *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR).*

[8] Arindam Das, S. Roy, U. Bhattacharya et S. K. Parui, «Document Image Classification with Intra-Domain Transfer Learning and Stacked Generalization of Deep Convolutional Neural Networks,» *arXiv:1801.09321.*

[9] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li et L. Fei-Fei, «ImageNet: A Large-Scale Hierarchical Image Database,» *IEEE Computer Vision and Pattern Recognition (CVPR), 2009.*

[10] M. Z. Afzal, S. Capobianco, M. I. Malik, S. Marinai, T. M. Breuel, A. Dengel et M. Liwicki, «Deepdocclassifier: Document classification with deep convolutional neural network,» *in 2015 13th International Conference on Document Analysis and Recognition.*

[11] A. Kölsch, M. Z. Afzal, M. Ebbecke et M. Liwicki, «Real-time document image classification using deep CNN and extreme learning machines,» *arXiv preprint arXiv:1711.05862, 2017.*

[12] Y. Xu, W. He, F. Yin et C. Liu, «Page Segmentation for Historical Handwritten Documents Using Fully Convolutional Networks,» *2017 14th IAPR International*

Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 541-546, doi: 10.1109.

[13] F. Simistira et al. , «ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts,» 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 1361-1370, doi: 10.1109/ICDAR.2017.223.

[14] L. Kang, J. Kumar, P. Ye, Y. Li and D. Doermann, "Convolutional Neural Networks for Document Image Classification," 2014 22nd International Conference on Pattern Recognition, Stockholm, 2014, pp. 3168-3172, doi: 10.1109/ICPR.2014.546.

[15] C. Tensmeyer et T. Martinez, «Analysis of Convolutional Neural Networks for Document Image Classification,» *arXiv:1708.03273.*

[16] L. Studer, M. Alberti, V. Pondenkandath, P. Goktepe, T. Kolonko, A. Fischer, M. Liwicki et Rolf Ingold, «A Comprehensive Study of ImageNet Pre-Training for Historical Document Image Analysis,» *arXiv:1905.09113,* 2019.

[17] J. Daniel, «L'annotation, le texte et le web,» [En ligne]. Available: https://ig.hypotheses.org/533.

[18] K. Zagoris, I. Pratikakis, A. Antonacopoulos, B. Gatos et N. Papamarkos, «Distinction between handwritten and machine-printed text based on the bag of visual words model,» *Pattern Recognition, vol. 47, no. 3, pp. 1051 – 1062, 2014.*

[19] K. Chen, H. Wei, M. Liwicki, J. Hennebert et R. Ingold, «Robust text line segmentation for historical manuscript images using color and texture,» *in 2014 22nd International Conference on Pattern Recognition, Aug 2014, pp. 2978–2983.*

[20] K. Chen, M. Seuret, M. Liwicki, J. Hennebert et R. Ingold, «Page Segmentation of Historical Document Images with Convolutional Autoencoders,» *1101-. 10.1109/ICDAR.2015.7333914..*

[21] C. Shin et D.S. Doermann, «Document image retrieval based on layout structural similarity,» *in IPCV, 2006, pp. 606–612.*

[22] Y. Byun et Y. Lee, «Form classification using dp matching,» in ACM Symposium on Applied Computing, New York, USA, 2000, pp. 1–4.

[23] T. Kochi et T. Saito, «User-defined template for identifying document type and extracting information from documents,» *in ICDAR, Sep 1999, pp. 127–130.*

[24] K. Collins-Thompson et R. Nickolov, «A clustering-based algorithm for automatic document separation,» 2002.

[25] J. Kumar, P. Ye et D. Doermann, «Structural similarity for document image classification and retrieval,» *Pattern Recognition Letters, vol. 43, pp. 119 – 126, 2014.*

[26] D. Dimmick, M. Garris et C. Wilson, «Nist structured form reference set of binary images (sfrs),» *[Online]. Available : (http://www.nist.gov/srd/nistsd2.cfm)*, 1991.

[27] A. W. Harley, A. Ufkes et K. G. Derpanis, «Evaluation of deep convolutional nets for document image classification and retrieval,» *in 13th International Conference on Document Analysis and Recognition (ICDAR), 2015. IEEE, 2015, pp. 991–995.*

[28] A. S. Razavian, H. Azizpour, J. Sullivan et S. Ca, «CNN Features off-the-shelf: An Astounding Baseline for Recognition,» *arXiv:1403.6382.*

[29] S. J. Pan et Q. Yang, «A Survey on Transfer Learning,» in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[30] S. Thrun et L. Pratt, «Learning to learn,» *Springer Science & Business Media, 2012.*

[31] A. Krizhevsky, «Learning multiple layers of features from tiny images,» *Technical report, University of Toronto, 2009.*

[32] S. Thrun et L. Pratt, «Machine Learning - Special Issue on Inductive Transfer,» *Springer, 1997.*

[33] F. Rosenblatt, «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,» *Psychological Review, 1958.*

[34] P. Paquet, «L'utilisation des réseaux de neurones artificiels en finance».

[35] R. Rojas, «Neural Networks: A Systematic Introduction,» *Springer-Verlag, 1996.*

[36] «Réseau de neurones Artificiels,» Wikipedia, [En ligne]. Available: https://fr.wikipedia.org/wiki/Réseau_de_neurones_artificiels.

[37] P. Hensman et D. Masko, «The Impact of Imbalanced Training Data for Convolutional Neural Networks,» *(https://www.kth.se/social/files/588617ebf2765401cfcc478c/PHensmanDMasko_dkand15.pdf).*

[38] MCGILL, «Apprentissage Automatique : Les réseaux de neurones,» Visité en Août 2020, [En ligne]. Available: https://thebrain.mcgill.ca/flash/capsules/pdf_articles/reseau_neurones.pdf.

[39] «Réseaux de Neurones,» Visité en Août 2020, [En ligne]. Available: https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf.

[40] J. Chen, X. Pan, R. Monga, S. Bengio et R. Jozefowicz, «Revisiting distributed synchronous SGD,» *arXiv preprint arXiv:1604.00981 [cs.LG], 2016.*

[41] L. Bottou, F. E. Curtis et J. Nocedal, «Optimization methods for large-scale machine learning,» *arXiv preprint arXiv:1606.04838 [stat.ML], 2016.*

[42] D. E. Rumelhart, G. E. Hinton et R. J. Williams, «Learning representations by back-propagating errors,» *Nature, vol. 323, no 6088, 8 octobre 1986, p. 533–536.*

[43] L. Bottou et O. Bousquet, «The Tradeoffs of Large-Scale Learning,» *In Advances in Neural Information Processing Systems, vol.20, 161-168,* 2008.

[44] «Momentum and Learning Rate Adaptation,» Visité en Août 2020, [En ligne]. Available: https://www.willamette.edu/~gorr/classes/cs449/momrate.html.

[45] «Typical CNN,» Visité en Août 2020, [En ligne]. Available: https://commons.wikimedia.org/wiki/File:Typical_cnn.png.

[46] «A comprehensive Guide to Convolutional Neural Network,» Visité en Août 2020, [En ligne]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[47] Y. LeCun, K. Kavukcuoglu et C. Farabet, «Convolutional networks and applications in vision,» In: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on (2010), pp. 253–256.

[48] «Réseau neuronal convolutif,» Visité en Août 2020, [En ligne]. Available: https://fr.wikipedia.org/wiki/Réseau_neuronal_convolutif.

[49] D. C. Cireşan, U. Meier et J. Schmidhuber, «Multi-column deep neural networks for image classification,» *Arxiv preprint arXiv:1202.2745, 2012.*

[50] «Dive into Deep Learning,» Visité en Août 2020. [En ligne]. Available: http://d2l.ai/chapter_convolutional-neural-networks/conv-layer.html.

[51] «Convolutional layer,» Visité en Août 2020, [En ligne]. Available: https://commons.wikimedia.org/wiki/File:Conv_layer.png.

[52] «Max Pooling,» Visité en Août 2020, [En ligne]. Available: https://commons.wikimedia.org/wiki/File:Max_pooling.png.

[53] «Convolutional Neural Networks : Step 3 : Flattening,» Visité en Août 2020, [En ligne]. Available: https://www.superdatascience.com/convolutional-neural-networks-cnn-step-3-flattening/.

[54] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella et J. Schmidhuber, «Flexible, High-performance convolutional neural networks for Image Classification,» *Arxiv preprint arXiv:1102.0183, 2011.*

[55] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy,

A. Khosla, M. Bernstein et al. , «Imagenet large scale visual recognition challenge,» *International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015. 1.*

[56] L. Blier, «A brief report of the heuritech deep learning,» meetup #5 (2016). Available at: https://heuritech.wordpress.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/].

[57] M. Huh, P. Agrawal et A. A. Efros, «"What makes ImageNet good for transfer learning?",» *arXiv preprint arXiv:1608.08614, 2016.*

[58] R. Fergus et M. D. Zeiler, «Visualizing and understanding convolutional networks,» *In European conference on computer visionSpringer, 2014, pp. 818–833.*

[59] K. P. Murphy, «Machine Learning: A Probabilistic Perspective,» *Cambridge: MIT Press. p. 247. ISBN 978-0-262-01802-9.*, 2012.

[60] H. Zulkifli, «Understanding Learning Rates and How It Improves Performance in Deep Learning,» *(21 January 2018) Towards Data Science. Retrieved 15 February.*

[61] N. Buduma et N. Locascio, «Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms,» *O'Reilly. p. 21. ISBN 978-1-4919-2558-4.*, 2017.

[62] L. N. Smith, «Cyclical Learning Rates for Training Neural Networks,» *arXiv:1506.01186.*

[63] I. Goodfellow, Y. Bengio et A. Courville, «Deep Learning,» *MIT Press, Cambridge, MA, 2016..*

[64] Yoshua Bengio, «Practical recommendations for gradient-based training of deep architectures,» *arXiv:1206.5533.*

[65] D. Masters et C. Luschi, «Revisiting Small Batch Training for Deep Neural Networks,» 2018.

[66] D. Randall Wilson et T. R. Martinez, «The general inefficiency of batch training for gradient descent learning,» *Neural Networks, 16(10):1429–1451, 2003.*

[67] Y. LeCun, L. Bottou, G. B. Orr et K-R. Müller, «Efficient backprop. In Neural Networks: Tricks of the Trade,» *pp. 9–48. Springer-Verlag, Berlin, 2012.*

[68] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy et P. T. P. Tang, «On large-batch training for deep learning: Generalization gap and sharp minima».

[69] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao et M. Ranzato, «Large scale distributed deep networks. In Advances in Neural Information Processing Systems,» *25, NIPS 2012, pp. 1223–1231, 2012.*

[70] D. Das, S. Avancha, D. Mudigere, K. Vaidynathan, S. Sridharan, D. Kalamkar , B. Kaul et P. Dubey, «Distributed deep learning using synchronous stochastic gradient descent,» *arXiv preprint arXiv:1602.06709.*

[71] A. Krizhevsky, «One weird trick for parallelizing convolutional neural networks,» *arXiv preprint arXiv:1404.5997 [cs.NE], 2014.*

[72] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia et K. He, «Accurate, large minibatch SGD: Training ImageNet in 1 hour,» *arXiv preprint arXiv:1706.02677 [cs.CV], 2017.*

[73] S. L. Smith, P.-J. Kindermans, C. Ying et Q. V. Le, «Don't Decay the Learning Rate, Increase the Batch Size».

[74] E. Hoffer, I. Hubara et D. Soudry, «Train longer, generalize better: closing the generalization gap in large batch training of neural networks».

[75] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting».

[76] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever et R.R. Salakhutdinov, «Improving neural networks by preventing co-adaptation of feature detectors,» *arXiv preprint arXiv:1207.0580, 2012.*

[77] Y. Simard, D. Steinkraus et J.C. Platt, «Best practices for convolutional neural networks applied to visual document analysis,» *In Proceedings of the Seventh International Conference on Document Analysis and Recognition, volume 2, pages 958–962, 2003.*

[78] C. Shorten et T. M. Khoshgoftaar, «A survey on Image Data Augmentation for Deep Learning,» *J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0.*

[79] F. Perez, C. Vasconcelos, S. Avila et E. Valle, «Data Augmentation for Skin Lesion Analysis,» *arXiv:1809.01442.*

[80] Webstorm, Visité en Mars 2019, [En ligne]. Available: https://www.jetbrains.com/fr-fr/webstorm/?fromMenu.

[81] Angular, Visité en Mars 2019, [En ligne]. Available: https://angular.io.

[82] Typescript, Visité en Mars 2019, [En ligne]. Available: https://www.typescriptlang.org.

[83] Angular, «Angular Material,» Visité en Mars 2019, [En ligne]. Available: https://material.angular.io.

[84] «Node.js,» Visité en Mars 2019, [En ligne]. Available: https://nodejs.org/en/.

[85] «Express.js,» Visité en Mars 2019, [En ligne]. Available: https://expressjs.com/fr/ .

[86] ejs, Visité en Août 2020, [En ligne]. Available: https://ejs.co.

[87] «Mongoose.js,» Visité en Mars 2019, [En ligne]. Available: https://mongoosejs.com.

[88] «Base de données relationnelle vs base de données non relationnelle,» Visité en Novembre 2020, [En ligne]. Available: https://www.oracle.com/fr/database/base-donnees-relationnelle-difference-non-relationnelle.html.

[89] «Pdf-Image,» Visité en Mars 2019, [En ligne]. Available: https://www.npmjs.com/package/pdf-image.

[90] «Tesseract.js,» Visité en Mars 2019, [En ligne]. Available: https://tesseract.projectnaptha.com.

[91] MongoDB, Visité en Août 2020, [Online]. Available: https://www.mongodb.com/3.

[92] «La Chartreuse de Parme, par l'auteur de "Rouge et noir" [Stendhal] (1783-1842),» Bibliothèque nationale de France, département Réserve des livres rares, RES-Y2-3659, [En ligne]. Available: https://gallica.bnf.fr/ark:/12148/btv1b86232971?rk=214.

[93] F. Simistira, M. Seuret , N. Eichenberger, A. Garz, M. Liwicki et R. Ingold, «DIVA-HisDB: A Precisely Annotated Large Dataset of Challenging Medieval Manuscripts,» *International Conference on Frontiers in Handwriting Recognition pp. 471-476,* 2016.

[94] «Histoire de la peinture en Italie, par M. Beyle [Stendhal] (1783-1842) Tome 1,» Bibliothèque nationale de France, département Réserve des livres rares, RES P-V-564 (1), [En ligne]. Available: https://gallica.bnf.fr/ark:/12148/bpt6k1525801n.

[95] «Armance, ou Quelques scènes d'un salon de Paris en 1827 [par Stendhal] (1783-1842),» Bibliothèque nationale de France, département Réserve des livres rares, RES-Y2-3658, [En ligne]. Available: https://gallica.bnf.fr/ark:/12148/btv1b8623293c?.

[96] Anaconda, Visité en Mars 2019, [En ligne]. Available: https://www.anaconda.com.

[97] Python, Visité en Mars 2019, [En ligne]. Available: https://www.python.org.

[98] Conda, Visité en Mars 2019, [En ligne]. Available: https://docs.conda.io/en/latest/.

[99] Jupyter Notebook, Visité en Mars 2019, [En ligne]. Available: https://jupyter.org.

[100] Markdown, Visité en Mars 2019, [En ligne]. Available: https://guides.github.com/features/mastering-markdown/.

[101] Keras, Visité en Mars 2019, [En ligne]. Available: https://keras.io.

[102 CNTK, Visité en Mars 2019, [En ligne]. Available: https://docs.microsoft.com/en-

] us/cognitive-toolkit/.

[103] Theano, Visité en Mars 2019, [En ligne]. Available: https://github.com/Theano/Theano.

[104] Tensorflow, Visité en Mars 2019, [En ligne]. Available: https://www.tensorflow.org.

[105] Matplotlib, Visité en Mars 2019, [En ligne]. Available: https://matplotlib.org.

[106] Flask, Visité en Mars 2019, [En ligne]. Available: https://flask.palletsprojects.com/en/1.1.x/.

[107] «Use Early Stopping to Halt the Training of Neural Networks At the Right Time,» Visité en Mars 2019, [En ligne]. Available: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/.

[108] Sergey Ioffe et C. Szegedy, «Batch normalization: Accelerating deep network training by reducing internal covariate shift,» *In Proceedings of 32nd International Conference on Machine Learning, ICML15, pp. 448–456, 2015.*

[109] S. Ioffe, «Batch renormalization: Towards reducing minibatch dependence in batch-normalized models,» *arXiv preprint arXiv:1702.03275 [cs.LG], 2017..*