République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique
Département Informatique

Mémoire de Master

Filière : Informatique

Spécialité : MIND

Thème :

UTILISATION DES RESEAUX BAYESIENS PROFONDS POUR LA FOUILLE DE DONNEES MASSIVES

Réalisé par :

AKRETCHE Kahina

MERAH Samia

Encadré par :

Mme. ALIANE Hassina

Soutenu le : xx/xx/xxxx devant le jury composé de :

M. XXXX xxxx (président)

Mme. XXXX xxxx

Mme. XXXX xxxx

Projet N° : 024 / 2020

2019-2020

À nos familles ;
À tous nos amis et proches.

Remerciements

C'est avec un grand plaisir que nous réservons cette page en signe de gratitude et de reconnaissance aux personnes ayant contribué de manière plus ou moins directe à ce mémoire.

Nous tenons tout d'abord à remercier dieu qui nous a donné le pouvoir et la volonté pendant 18 ans consécutifs jusqu'à ce jour de soutenance du projet de fin d ?études, tout en lui demandant de nous apporter d'avantage pour continuer.

Ainsi, on tient à remercier Mme. ALIANE Hassina pour son encadrement, son orientation et ses efforts afin qu'on puisse mener à bien ce projet.

Nos sincères remerciements aux membres de jury de nous faire honneur de juger cet humble mémoire.

Nous adressons notre profonde gratitude à tout l'ensemble du corps professionnel et administratif du département Informatique ayant contribué à notre formation.

En fin, nos remerciements vont à nos familles et à nos amis et à tous ce qui ont contribué de près ou de loin à la réalisation de ce mémoire.

Kahina & Samia

Résumé

Résumé

Après les nombreux hivers de l'Intelligence artificielle de la seconde partie du XXème, la discipline connaît une nouvelle jeunesse à partir des années 90, grâce notamment aux travaux de Yann Lecun, Yoshua Bengio et Geoffrey Hinton (pères du deep learning) dans le domaine des réseaux de neurones appliqués à la vision par ordinateur (computer vision). Dès lors, les réseaux de neurones ont permis de nombreuses avancées dans des champs d'étude très variés allant de la finance à la logistique en passant par la mécanique des fluides. Cependant, malgré les très bons résultats obtenus par les réseaux de neurones, une certaine méfiance persiste encore vis-à-vis de ces solutions. Deux problèmes majeurs ralentissent en effet leur développement : leur manque d'interprétabilité et la difficulté à associer un degré de confiance/certitude à une prédiction. Très souvent, il est impossible ou difficile d'obtenir une connaissance certaine. Dans la vie réelle, on se trouve fréquemment dans des situations de prise de décisions dans lesquelles on ne dispose que d'informations incomplètes ou incertaines. Ces problématiques constituent un frein à la mise en production de ces solutions et à leur implantation en dehors du domaine académique. C'est au problème concernant le degré de confiance des prédictions que se propose de répondre le réseau profond bayésien, notion développée dans ce mémoire.

Mots Clés

deep learning, réseaux de neurones, certitude, prise de décisions, réseau profond bayésien.

Abstract

After many winters of the Artificial Intelligence during the second part of the XXth, the discipline experienced a new youth from the 90s, thanks in particular to the work of Yann Lecun, Yoshua Bengio and Geoffrey Hinton (fathers of deep learning) in the field of neural networks applied to computer vision. Since then, neural networks have enabled numerous advances in a wide variety of fields of study, from finance to logistics and fluid mechanics. However, despite the remarkable results obtained by neural networks, a certain mistrust still persists towards these solutions. Two major problems slow down their development : their lack of interpretability and the difficulty to associate a degree of confidence/certainty to a prediction. Very often, it is impossible or difficult to obtain a certain knowledge. In real life, we frequently find ourselves in decision-making situations (business intelligence) where we have incomplete or uncertain information. These problems prevent the production of these solutions and their implementation outside the academic field. It is to the problem concerning the degree of confidence of predictions that the Deep Bayesian Network, a concept developed in this thesis, proposes to respond.

Key words

deep learning, neural networks, certainty, decision-making, deep bayesian network.

Table des matières

Liste des abréviations

Table des figures

Liste des tableaux

Introduction générale

Ces dernières années ont vu la convergence de trois révolutions : une révolution technologique, une révolution des données et une révolution des usages. Ensemble, ces trois révolutions constituent ce que l'on nomme « Big Data ».

Littéralement, ce terme signifie mégadonnées, grosses données ou encore données massives. Ils désignent un ensemble très volumineux de données qu'aucun outil classique de gestion de base de données ou de gestion de l'information ne peut vraiment travailler. En effet, nous procréons environ 2,5 trillions d'octets de données tous les jours. Ce sont les informations provenant de partout : messages que nous nous envoyons, vidéos que nous publions, informations climatiques, signaux GPS, enregistrements transactionnels d'achats en ligne et bien d'autres encore. Ces données sont baptisées Big Data ou volumes massifs de données. Les géants du Web, au premier rang desquels Yahoo (mais aussi Facebook et Google), ont été les tous premiers à déployer ce type de technologie.

Ce concept regroupe une famille d'outils qui répondent à une triple problématique dite règle des 3V. Il s'agit notamment d'un **Volume** de données considérable à traiter, une grande **Variété** d'informations (venant de diverses sources, non-structurées, organisées, Open...), et un certain niveau de **Vélocité** à atteindre, autrement dit de fréquence de création, collecte et partage de ces données.

Ces trois facteurs sont une composante essentielle du Big Data. Il faut nécessairement les considérer pour gérer, analyser et traiter la masse considérable d'informations circulant chaque jour. Le Big Data se présente comme une évolution à laquelle personne ne peut se soustraire.

La quantité des données produite augmente constamment. En raison de leur quantité et de leur volume, les outils classiques de gestion et d'analyse sont incapables de traiter convenablement ces données, ce qui rend leur traitement de plus en plus difficile à gérer avec les outils actuels.

C'est là que l'intelligence artificielle intervient. Seuls des algorithmes sophistiqués de data mining sont aujourd'hui capables de traiter autant d'informations en instantanée.

L'intelligence artificielle va être utilisée pour extraire du sens, déterminer de meilleurs résultats, et permettre des prises de décisions plus rapides à partir de sources Big Data massives.

Aujourd'hui, l'usage des techniques de l'intelligence artificielle (machine learning, deep learning), des systèmes experts et des technologies analytiques en combinaison avec le Big Data se présentent comme l'évolution naturelle de ces deux disciplines. La convergence est inéluctable.

Comme le dit le vieil adage « **trop d'informations tuent l'information** ». Il s'agit en fait du principal problème avec les mégadonnées. La quantité énorme des informations est un des obstacles. L'autre obstacle provient évidemment du niveau de certitude qu'on peut avoir sur une donnée.

Les outils d'apprentissage profond (deep learning) ont acquis une attention considérable dans l'apprentissage machine appliqué. Toutefois, ces outils ne tiennent pas compte de l'incertitude du modèle. En comparaison, les modèles bayésiens offrent un cadre mathématiquement fondé à la raison de l'incertitude du modèle.

A ce propos, on peut distinguer deux types d'écoles dans le domaine prédictif à savoir le deep learning et les réseaux bayésiens. Ces deux secteurs bien qu'ils soient distincts se rejoignent finalement de plus en plus. De plus, ils peuvent être utilisés en simultanéité de manière vertueuse et intelligente pour mener à bien un projet, c'est ce que l'on nomme **Les réseaux Bayésiens Profonds** (Bayesian Deep Networks en anglais).

L'objectif de notre projet est d'implémenter les réseaux bayésiens profonds en proposant une approche adéquate. Pour cela, nous utilisons les réseaux de neurones convolutifs en ajoutant l'aspect bayésien dans le but de classifier un jeu de données d'images.

Ce mémoire comporte trois chapitres, définis comme suit :

— **Chapitre 1 « L'analyse du big data » :** ce chapitre porte sur les outils d'analyse du big data, il offre une vision sur l'apprentissage automatique en se focalisant sur l'apprentissage profond et ses notions importantes et donne un petit aperçu sur les réseaux bayésiens.

— **Chapitre 2 « Réseaux bayésiens profonds » :** ce second chapitre sera consacré aux réseaux bayésiens profonds ainsi qu'à l'implémentation de l'approche de Monte Carlo Dropout dans l'apprentissage profond bayésien en vue de prouver son efficacité pour explorer les valeurs du Big Data

— **Chapitre 3 « Implémentation et résultats expérimentaux » :** ce dernier chapitre expose les résultats des différentes expérimentations réalisées.

Enfin, avant de clôturer ce mémoire en parlant des perspectives possibles, nous dressons une conclusion générale du projet.

CHAPITRE 1

L'ANALYSE DU BIG DATA

1.1 Introduction

Il sera question dans ce chapitre de définir le concept d'analyse du big data ainsi que les approches du data mining d'analyse utilisées. Nous commençons par l'apprentissage automatique et ses différentes méthodes. Nous mettons en exergue le Deep Learning avec les réseaux de neurones et plus précisément les réseaux convolutifs (CNN) que nous utilisons dans ce travail. Nous citons par la suite quelques limites de ces approches en introduisant le rôle des réseaux bayésiens.

1.2 Data mining (Fouille de données)

Le volume des données circulant sur le Web, ou stockées par les entreprises est en croissance continue. Afin de pouvoir exploiter cette richesse, il est nécessaire d'extraire des connaissances à partir de très grands volumes d'informations.

La nature de l'analyse de très grands volumes de données dépend de la nature et de la structure des Big Data, que l'on appelle aussi « analytique », traduction du terme anglo-saxon « analytics ». Ces différentes analyses mettront en œuvre divers algorithmes relevant de la fouille de données (Data Mining) (**Bernard ESPINASSE et Patrice BELLOT, 2017**).

1.2.1 Définition

Le data mining ou fouille de données est l'ensemble des méthodes scientifiques destinées à l'exploration et l'analyse de grandes quantités de données informatiques en vue de détecter des profils-type, des comportements récurrents, des règles, des liens, des tendances inconnues, des structures particulières restituant de façon concise l'essentiel de l'information utile pour l'aide à la décision (**Mat, 2012**).

Le data mining permet l'extraction des données selon plusieurs algorithmes. Ces algorithmes couvrent :

— La classification, est une méthode qui permet de regrouper des objets (personnes, intérêt...) en groupes, ou familles de sorte que les objets d'un même groupe se ressemblent le plus possible, et ceux de groupes distincts diffèrent le plus possible.

— Le clustering, regroupement automatique d'individus au sein d'un certain nombre de classes à priori inconnues.

— La régression qui est une estimation automatique d'une fonction mathématique permettant de faire correspondre des entrées, par exemple des vecteurs décrivant des individus et des sorties, ou encore des classes ou des valeurs numériques.

— L'analyse d'association, entre individus ou entre variables.

— L'analyse de réseaux, ou graphes (**Corentin HARDY, Avril 2019**).

1.3 Machine Learning (Apprentissage automatique)

Le Machine Learning est une branche de l'Intelligence Artificielle qui permet à une machine d'analyser un système, de comprendre pas à pas son fonctionnement et comme résultat d'effectuer ou simuler différentes tâches de ce système. L'algorithme d'apprentissage a pour objectif d'apprendre le fonctionnement du système étudié de manière active. Il connait les entrées possibles du système et compose des séquences qu'il soumet au système (requêtes) pour observer ses réponses (séquences autorisée/refusée, valeurs renvoyées, etc.) (**Ork, 2012**).

D'autre part, nous pouvons définir le terme Machine Learning comme un ensemble d'algorithmes qui permettent d'apprendre le fonctionnement d'un système en observant régulièrement les tâches qu'il réalise, puis prédire son comportement et ses décisions.

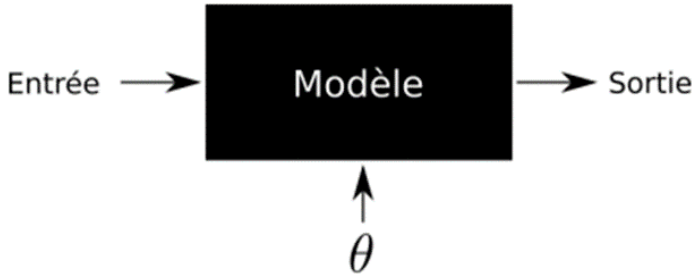


Figure 1.1 – Modèle d'apprentissage vu comme une boîte noire

1.3.1 Méthodes et mécanismes du Machine Learning

Les méthodes d'apprentissage automatique s'organisent en trois types : méthodes supervisées, semi-supervisées, et non supervisées.

— L'apprentissage supervisé est traditionnellement synonyme de la classification. La supervision dans l'apprentissage vient du fait que les observations de la base d'apprentissage sont étiquetées.

— L'apprentissage non supervisé est essentiellement synonyme de clustering. Le processus d'apprentissage est non supervisé puisque les observations d'entrée ne sont pas étiquetées.

— L'apprentissage semi-supervisé est une classe de techniques qui font usage des deux types d'observations, étiquetées et non étiquetées, lors de l'apprentissage d'un modèle (**Corentin HARDY, Avril 2019**).

1.4 Réseaux de neurones (NN)

Initialement conçus dans le but de modéliser mathématiquement le traitement de l'information des neurones biologiques du cortex humain, leur rapprochement aujourd'hui n'importe plus autant et c'est leur efficacité à construire des modèles très complexes et non linéaires qui fait leur succès. Les réseaux de neurones (ou en anglais Neural Networks) sont des systèmes composés de processeurs élémentaires appelés neurones artificiels, généralement répartis en plusieurs couches interconnectées et fonctionnant en parallèle. Chaque neurone reçoit en entrée des signaux plus ou moins forts (les activations) provenant d'autres neurones, et en fonction de ces activations et de l'importance (le poids) qu'il donne à chacun d'eux, il émet à son tour un signal (s'activer) ou pas (**Rémi Connesson, 2018**).

1.4.1 Le neurone formel

Un neurone formel (ou artificiel), introduit par McCulloch et Walter Pitts en 1943 (**McCulloch, W.S. & Pitts, 1943**), est une modélisation mathématique d'un neurone biologique. Il consiste en une fonction mathématique à appliquer à un signal et renvoyant une valeur d'activation.

1.4.2 Le perceptron

Franck Rosenblatt introduit en 1958 un algorithme d'apprentissage automatique basé sur le neurone formel appelé le perceptron (**Rosenblatt, 1958**).

Le perceptron est formé d'une première couche d'unités (ou neurones) qui permettent de « lire » les données : chaque unité correspond à une des variables d'entrée. On peut rajouter une unité de biais qui est toujours activée (elle transmet 1 quelles que soient les données). Ces unités sont reliées à une seule et unique unité de sortie, qui reçoit la somme des unités qui lui sont reliées, pondérée par des poids de connexion. Pour p variables $x_1, x_2, ..., x_P$, la sortie reçoit donc $w_0 + \sum_{j=1}^{p} w_j x_j$.

L'unité de sortie applique alors une fonction d'activation à cette sortie.

Un perceptron prédit donc grâce à une fonction de décision f définie par :

$$f(x) = a(w_0 + \sum_{j=1}^{p} w_j x_j) \tag{1.1}$$

(**Chloé-Agathe Azencott, 2019**).

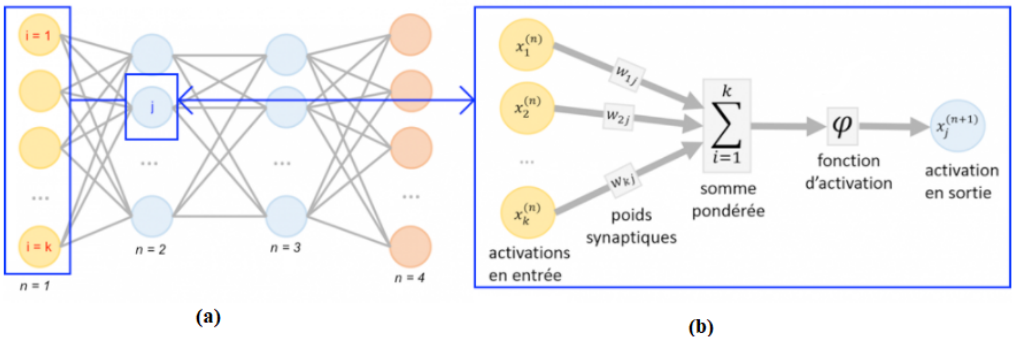


FIGURE 1.2 – (a) un NN organisé en 4 couches, (b) le mécanisme d'activation d'un neurone (**1**)

1.4.3 Le perceptron multi-couches

Comme son nom l'indique le perceptron multi-couches (ou Multilayer Perceptron, souvent abrégé « MLP », en anglais) est un réseau formé de plusieurs couches intermédiaires. Il consiste en un empilement de plusieurs neurones artificiels formant une couche cachée. Chaque caractéristique d'entrée est connectée à chaque neurone artificiel de la couche suivante. De même, les neurones artificiels de la couche cachée sont tous connectés aux neurones de couche de sortie (**2**).

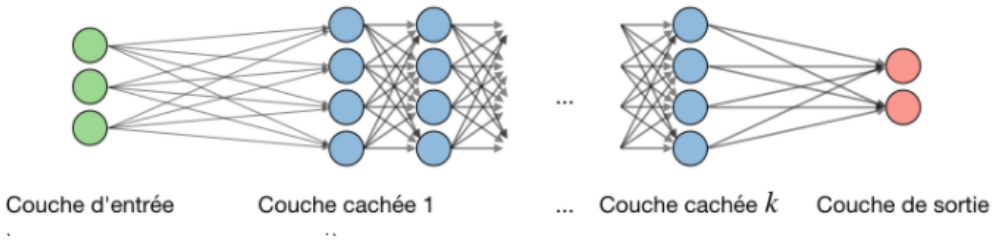


FIGURE 1.3 – Un exemple de perceptron-multicouches constitué de k couches (**3**)

1.5 Deep Learning

1.5.1 Définition

Deep Learning ou apprentissage profond, est une branche du Machine Learning particulièrement adaptée à l'apprentissage de données complexes en vue de réaliser des modèles avancés supervisés ou non supervisés (**STEPHENE TUFFERY, 2019**).

Le perceptron multi-couches est l'exemple par excellence d'un modèle d'apprentissage profond. Habituellement, un réseau de neurones est considéré comme profond, s'il est composé d'au moins quatre couches (c'est-à-dire trois couches cachées + une couche de sortie) (**2**).

Celles-ci permettent de décomposer de manière hiérarchique le contenu d'une donnée complexe comme de la voix ou une image pour la classifier ensuite : identifier des mots pour la voix ou associer des tags descriptifs à des images. Le Deep Learning sert le plus souvent à reconnaître le langage, l'écriture et les images mais il peut aussi avoir d'autres usages dans les outils d'aide à la décision (**Olivier Ezratty, Octobre 2017**).

1.5.2 Rétropropagation

La rétropropagation du gradient de l'erreur (ou backpropagation) est un algorithme d'optimisation permettant d'ajuster les paramètres d'un réseau de neurones multicouches pour mettre en correspondance des entrées et des sorties référencées dans une base d'apprentissage.

Pour pouvoir entraîner ces systèmes, il faut savoir comment ajuster les paramètres de chaque couche de neurones. Pour chaque exemple de données (groupe d'observations de l'ensemble d'apprentissage), l'entrée est donnée au modèle afin d'obtenir la sortie, celle-ci est comparée avec le résultat attendu pour calculer l'erreur, La rétropropagation permet de calculer le gradient de l'erreur pour chaque neurone, de la dernière couche vers la première (rétropropagation). Cela permet de corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs. Ainsi, les poids qui contribuent à engen-

drer une erreur importante se verront modifiés de manière plus significative que les poids qui ont engendré une erreur marginale (**4**).

La dérivée par rapport à chaque coefficient w est calculée en utilisant la règle de la chaîne ou chain rule (voir figure 1.4.). Chaque coefficient est mis à jour comme suit :

$$w \leftarrow w - a\frac{\partial L(x,y)}{\partial w} \tag{1.2}$$

avec L l'erreur, w le coefficient, x l'entrée, y la sortie et a le taux d'apprentissage.

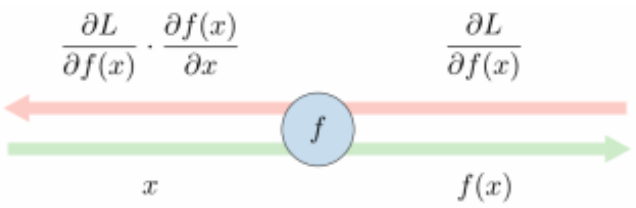


FIGURE 1.4 – Règle de la chaîne

Actualisation des coefficients : Dans un réseau de neurones, les coefficients sont actualisés comme suit :

— Étape 1 : Prendre un groupe d'observations appartenant aux données d'entrainement

— Étape 2 : Réaliser la propagation avant pour obtenir le loss (valeur de la fonction de perte) correspondant.

— Étape 3 : Effectuer une rétropropagation du loss pour obtenir les gradients.

— Étape 4 : Utiliser les gradients pour actualiser les coefficients du réseau

1.5.3 Sur-apprentissage

Un réseau de neurones apprend grâce à des exemples (jeux d'entraînements) qui lui sont soumis. Le but de cet apprentissage est de permettre au réseau de tirer de ces exemples des généralisations et de pouvoir les appliquer à de nouvelles données par la suite.

On parle de sur-apprentissage (le terme anglais est overfitting) quand un modèle a trop appris les particularités de chacun des exemples fournis en exemple. Il présente alors un taux de succès très important sur les données d'entraînement (pouvant atteindre jusqu'à 100%, au détriment de ses performances générales réelles, et sur de nouvelles données qu'il n'a pas encore vu, sa prédiction sera totalement biaisée (**5**).

1.5.4 Les différentes architectures du Deep Learning

Bien qu'il existe un grand nombre de variantes d'architectures pour l'apprentissage profond, il n'est pas toujours possible de comparer les performances de toutes les architectures, car elles ne sont pas toutes évaluées sur les mêmes ensembles de données. Le Deep Learning

est un domaine à croissance rapide, et de nouvelles architectures, variantes ou algorithmes apparaissent toutes les semaines.

1.6 Les réseaux de neurones convolutifs

Les réseaux de neurones convolutifs sont un type particulier de réseau de neurones. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network. Le nom de ces réseaux vient du fait que leur fonctionnement est basé sur l'utilisation d'une opération mathématique linéaire appelée convolution (**Ian Goodfellow et al, 2016**).

Chaque couche - dite de convolution - balaye l'ensemble de la couche précédente en appliquant à chaque petite région un même traitement local. Dans le cas d'un texte, les régions sont les n-grammes de la phrase. Pour une image, ils comportent deux parties bien distinctes, en entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

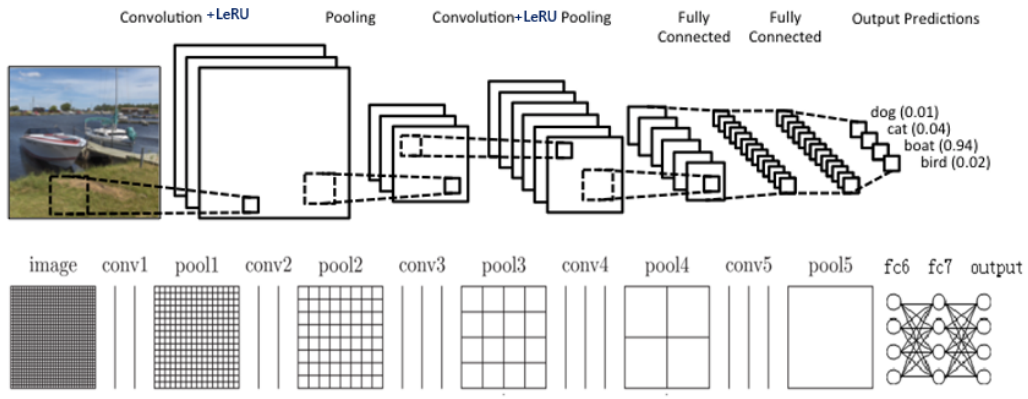


FIGURE 1.5 – Architecture standard d'un réseau de neurone convolutif (**6**)

1.6.1 L'opération de convolution

Les réseaux de neurones classiques ne s'adaptent pas parfaitement aux images complètes. Par exemple, certaines bases de données (comme CIFAR-10) ont des images de taille 32 x 32 x 3 (32 pixels de largeur, 32 de hauteur, 3 canaux de couleurs), de sorte qu'un seul neurone entièrement connecté dans une première couche cachée d'un réseau de neurones ordinaire aurait 32 * 32 * 3 = 3072 poids. Si cette quantité semble toujours gérable dans ce cas de figure, il est clair que cette structure entièrement connectée ne s'adapte pas aux images plus grandes. En effet, avec une image de taille plus commune, par exemple 200 x 200 x 3, ceci conduirait à avoir des neurones avec 200 * 200 * 3 = 120 000 poids. Avec plus de neurones, ces paramètres s'additionneraient très rapidement. Cette connectivité totale prend beaucoup de temps et le nombre considérable de paramètres conduirait sans doute à un surapprentissage (**Andrej Karpathy, 2019**).

1.6.2 Architecture CNN

Une architecture CNN est formée par un empilement de couches de traitement indépendantes :

— La couche de convolution (CONV) est la la composante clé d'un réseau de neurones convolutif. Elle effectue la majeure partie des lourdes tâches de calcul. La couche de convolution reçoit en entrée plusieurs images, et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent aux caractéristiques (features) que l'on souhaite retrouver dans les images. La sortie résultante est appelée carte de caractéristiques (en anglais activation map ou feature map), et nous indique où se situent ces features. Une couche convolutive est constituée de plusieurs filtres (ou noyaux) de convolution à appliquer sur une matrice d'entrée (une image ou une feature map précédente). Les noyaux des filtres désignent les poids de la couche de convolution. Ils sont initialisés puis mis à jour par **la descente du gradient** (**Radford et al, 2015**).
La convolution peut être ajustée selon certains paramètres, dont le nombre et la taille de filtres, le stride et le padding .Le stride est un paramètre indiquant de combien de pas on se déplace entre deux calculs de la convolution, et le padding indique combien de rangées de 0 on ajoute autour de l'entrée (Cette marge permet de contrôler la dimension spatiale du volume de sortie).

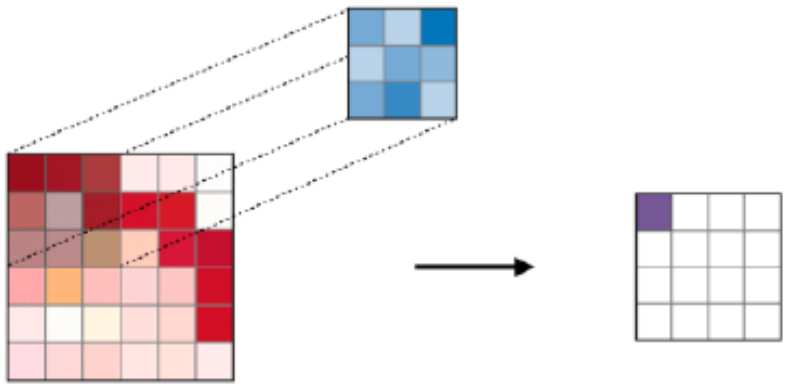


FIGURE 1.6 – Opération de convolution En rouge : image en input. En bleu : le filtre de convolution. En violet : résultat de l'application du filtre sur la partie de l'image sélectionnée (**7**)

— La couche d'activation : Il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie.
La fonction d'activation d'unité linéaire rectifiée ou couche ReLU pour Rectified Linear Unit Layer est certainement la plus utilisée dans les CNN. C'est une couche dite de correction qui force les neurones à retourner des valeurs positives (**Bengio et al.,**

2014).

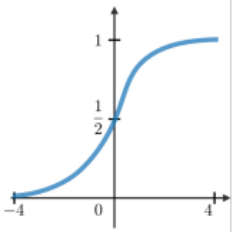
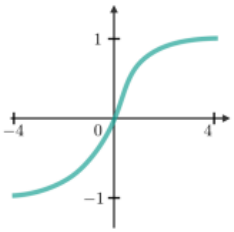
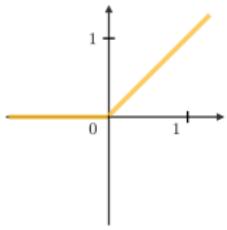
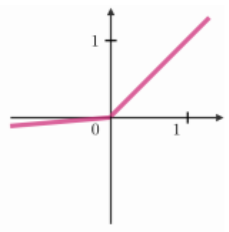
Sigmoïde	Tanh	ReLU	Leaky ReLU
$g(x) = \frac{1}{1+e^{-x}}$	$g(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$g(x) = \max(0,x)$	$g(x) = \max_{\epsilon \ll 1}(\epsilon x, x)$
			

FIGURE 1.7 – Types de fonctions d'activation (**7**)

— La couche de pooling (POOL), qui permet de compresser l'information en réduisant la taille de l'entrée intermédiaire (souvent par sous-échantillonnage), réduisant ainsi la quantité de paramètres et de calcul dans le réseau. Il est fréquent de trouver dans les architectures de CNN des couches POOL insérées périodiquement entre deux couches convolutives successives. Il existe plusieurs types de pooling, les plus populaires sont le max-pooling et l'average-pooling.

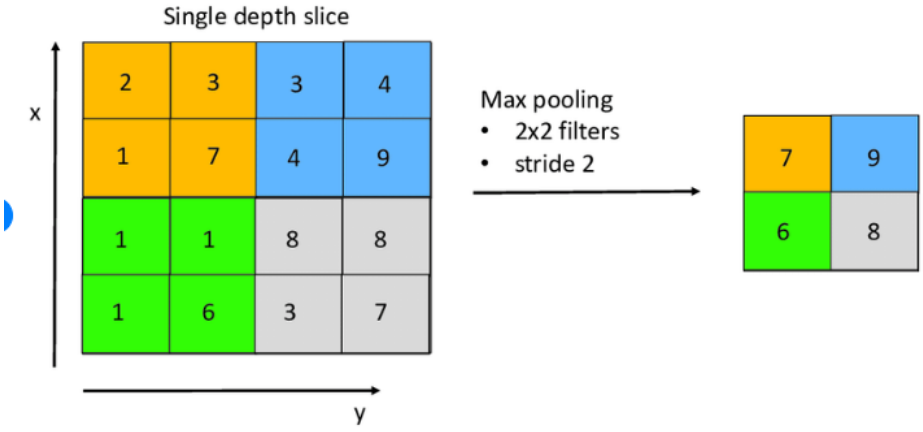


FIGURE 1.8 – Pooling avec un filtre 2x2 et un pas de 2 (**Giorgio Roffo,2017**)

— La couche "entièrement connectée" (en anglais fully connected layer) (FC) s'applique sur une entrée préalablement aplatie où chaque entrée est connectée à tous les neurones. Les couches de fully connected sont typiquement présentes à la fin des architectures de CNN et peuvent être utilisées pour optimiser des objectifs tels que les scores de classe.

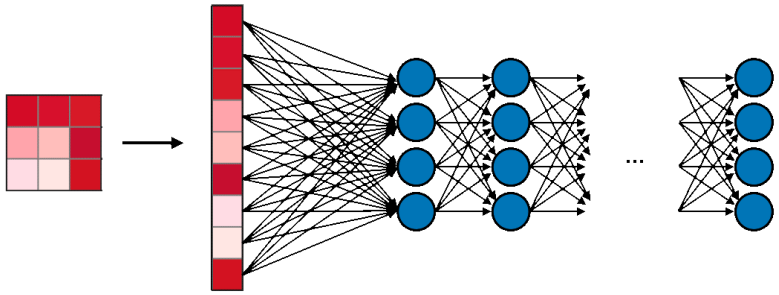


FIGURE 1.9 – Exemple de couche fully connected CNN (**7**)

La mise à plat (Flattening)

Il consiste simplement à mettre à bout toutes les entrées que nous avons pour en faire un (long) vecteur.

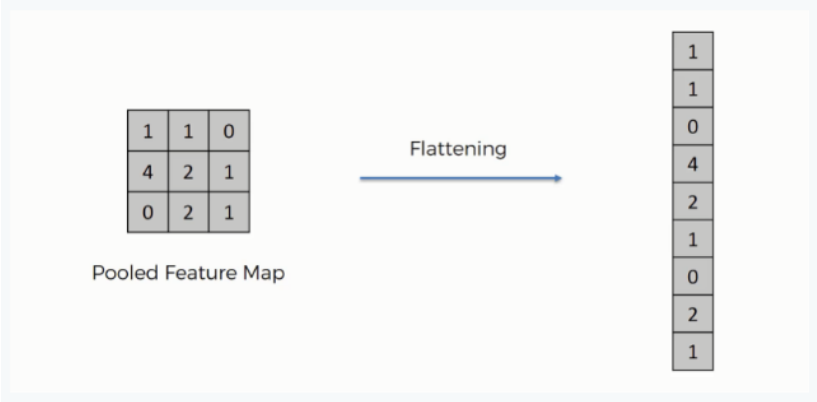


FIGURE 1.10 – Mise à plat des images (**8**)

— La couche de perte (LOSS) spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La fonction « Softmax » généralement utilisée pour l'optimisation de réseau de classification d'images, elle permet de calculer la distribution de probabilités sur les classes de sortie.

1.7 L'analyse des données du big data

A la lumière du volume important de données du big data il est nécessaire d'utiliser des moyens d'analyse différents des méthodes d'analyse traditionnelles.

Tout d'abord l'utilisation de l'Intelligence Artificielle ou IA qui peut se définir comme une discipline scientifique relative au traitement des connaissances et au raisonnement, dans le but de permettre à une machine d'exécuter des fonctions normalement associées à l'intelligence humaine : compréhension, raisonnement, dialogue, adaptation, apprentissage, etc. Les

programmes d'IA apprennent à partir des données afin de répondre intelligemment aux nouvelles données et ainsi fournir des résultats correspondants.

Un des moyens d'application de l'IA est l'apprentissage automatique et l'apprentissage profond. Grâce à l'apprentissage automatique, les ordinateurs apprennent sans être explicitement programmés. Lorsque nous achetons en ligne, l'apprentissage automatique permet de recommander d'autres produits qui pourraient nous intéresser en fonction des produits que nous avons déjà achetés. Lorsque nous utilisons notre carte de crédit, l'apprentissage automatique compare la transaction à une base de données de transactions telles que la zone géographique habituelle de transactions et ainsi permet la détection des fraudes. Lorsque notre voiture nous indique la durée d'un trajet jusqu'à notre domicile, l'apprentissage automatique permet de déterminer où se trouve notre domicile. L'apprentissage automatique est un sous domaine de l'intelligence artificielle. Le Deep Learning est lui-même une sous-catégorie de l'apprentissage automatique. L'exemple d'application le plus commun est la reconnaissance visuelle. Par exemple, un algorithme va être programmé pour détecter certains visages depuis les images en provenance d'une caméra. Suivant la base de données attribuée, il pourra repérer un individu recherché dans une foule, détecter le taux de satisfaction à la sortie d'un magasin en détectant les sourires, etc. Un ensemble d'algorithmes pourra également reconnaître la voix, le ton, l'expression d'un questionnement, d'une affirmation ainsi que les mots d'une conversation.

Pour résumer, le volume de données composant le big data peut être considéré comme difficilement exploitable sans un support informatique. L'IA est donc l'intelligence qui permet une exploitation efficace du big data, et l'apprentissage automatique et profond font partie des techniques qui facilitent l'analyse du big data (**9**).

1.8 Limites du machine learning dans le Big Data

Les algorithmes de machine Learning ne sont pas non plus une boite noire magique qui permet de tout deviner et qui s'adapte à tout. Si l'on souhaite faire des prédictions de bonne qualité, il y a certaines choses à prendre en compte.

Tout d'abord, il est nécessaire que les résultats que l'on souhaite prédire soient différentiables. Les algorithmes ne jouent que 20% dans la qualité des prédictions, les 80% autres sont dus à la qualité des données.

Dans le monde réel, les choses sont bien plus complexes, on peut avoir des imprécisions sur les caractéristiques, des erreurs dans la classification des données ou beaucoup d'autres facteurs qui rendent les prédictions moins évidentes. Une autre limite du Machine Learning est que les algorithmes sont incapables d'extrapoler les grands volumes de données de manière fiable. Il est donc nécessaire de faire des prédictions uniquement sur le même domaine de données que celui utilisé pour l'apprentissage. Les algorithmes de machine Learning ne permettent donc pas d'apprendre de nouvelles choses mais seulement d'automatiser des choses connues

ou de mettre en évidence des relations. Dans ce contexte, il est essentiel que les méthodes d'apprentissage puissent suivre le rythme des données, non seulement en termes de volume, mais aussi de la vitesse à laquelle elles sont générées et traitées, afin d'être utiles.

Les méthodes classiques de machine learning ont été récemment adaptées aux propriétés du Big Data en appliquant la philosophie Deep learning. Cette dernière vise à produire des représentations abstraites des caractéristiques observées, qui sont organisés en couches ; plus la couche est élevée, plus le niveau d'abstraction est élevé. Bien que le Deep Learning ait été appliqué dans une panoplie de domaines et ait donné des résultats appréciables, il a certains inconvénients. En fait, les couches superposées cachées sont considérées comme une boîte noire dont la taille et les paramètres sont définis empiriquement. De plus, la signification de chaque variable latente (cachée/non observée), dans une couche cachée, est perdue et l'interprétabilité de la représentation des caractéristiques abstraites est pratiquement impossible (**Hasna Njah et al, 2016**).

Construire des modèles d'apprentissage automatique à partir de grandes masses de données (Big Data) nécessite le développement de nouveaux types d'algorithmes. La plupart des algorithmes d'apprentissage automatique ne passent pas à l'échelle (**Olivier Ezratty, Octobre 2017**), elles ne sont pas suffisamment performantes pour exploiter pleinement la valeur du Big Data, sa variabilité et sa véracité rendent le processus de machine learning perplexe. Le volume de données est trop large pour des analyses complètes, et les corrélations et relations entre ces données sont trop importantes pour que les analystes puissent tester toutes les hypothèses afin de dégager une valeur de ces données. Les outils d'apprentissage profond ont acquis une attention considérable dans l'apprentissage machine appliqué. Toutefois, ces outils ne tiennent pas compte de l'incertitude du modèle, c'est là où les réseaux bayésiens semblent idéaux pour exploiter les opportunités cachées du Big Data.

1.9 Les réseaux bayésiens

1.9.1 C'est quoi un réseau bayésien ?

On peut définir un réseau bayésien par un modèle graphique regroupant au sein d'un même formalisme la théorie des graphes et celle des probabilités afin de fournir des outils efficaces autant qu'intuitifs pour représenter une distribution de probabilités jointe sur un ensemble de variables aléatoires. Ce formalisme très puissant permet une représentation intuitive de la connaissance sur un domaine d'application donné et facilite la mise en place de modèles performants et clairs. La représentation de la connaissance se base sur la description, par des graphes, des relations de causalité existant entre des variables décrivant le domaine d'étude. A chaque variable est associée une distribution de probabilités locale quantifiant la relation causale (**David Bellot, 2002**).

1.9.2 Pourquoi les réseaux bayésiens ?

Une des grandes problématiques de notre époque est de traiter la grande quantité des données qui est mise à notre disposition (notamment grâce à l'informatique) pour en extraire de l'information. Il serait donc intéressant d'avoir un (ou plusieurs) modèle(s) effectuant le lien entre les observations et la réalité pour un objectif précis, et cela, même lorsque les observations sont incomplètes et/ou imprécises.

Imaginons un statisticien qui veut analyser un tableau de mesures pour une population donnée. Il se retrouve face à une immense masse d'informations de laquelle il doit extraire de la connaissance ! Il va donc essayer de retrouver les relations pertinentes entre des variables ou des groupes de variables. L'utilisation des réseaux bayésiens va lui permettre d'obtenir une représentation compacte de ces ensembles de dépendances grâce à la notion de probabilités conditionnelles, à partir de laquelle il lui sera plus simple de raisonner.

Les réseaux bayésiens permettent donc de transformer en modèle interprétable la connaissance contenue dans des données.

Les réseaux probabilistes sont également une représentation du savoir incertain plus flexible que les systèmes à base de règles. Par exemple, en médecine, une même combinaison de symptômes peut être observée pour différentes maladies.

Prenons l'exemple du corps humain, système complexe à volonté. Lorsqu'un individu est malade, nous allons observer pour celui-ci un certain nombre de grandeurs (tension, fièvre, etc). En fonction de ces différentes observations et de sa connaissance a priori le médecin va donner son diagnostic. Ce diagnostic est ici évalué en fonction d'un nombre restreint de paramètres, or il se peut que deux individus aient la même forme (les mêmes valeurs pour toutes les grandeurs observées) et que l'un d'eux soit malade, tandis que l'autre est sain.

Les modèles probabilistes ont cet avantage de fournir systématiquement une probabilité à chaque état (ici sain ou malade). Celle-ci peut alors être considérée comme un indice de confiance dans le résultat (par exemple, cet individu a 85% de chance d'être malade). Bien sûr un tel diagnostic n'est pas satisfaisant d'un point de vue éthique pour décider d'administrer ou non un traitement.

Pour résumer :

— L'aspect graphique des modèles bayésiens permet de représenter les relations entre les attributs clairement et intuitivement.

— Leurs orientations (si elles existent) peuvent représenter des relations de cause à effet.

— Les modèles probabilistes sont capables de gérer l'incertain et l'imprécis (**Olivier Francois, 2006**).

1.10 Formalisme mathématique

1.10.1 Définition formelle (Réseaux bayésiens)

Un réseau bayésien $B = (G, \theta)$ est défini par :

— une structure $G = (V, E)$ qui est un graphe orienté sans circuit (DAG : Directed Acyclic Graph) où V est l'ensemble des nœuds qui représentent un ensemble de variables aléatoires $X = (X_1, ..., X_n)$ et E est l'ensemble des arcs,

— et des paramètres $\theta = [P(X_i/Pa(X_i))]$ qui sont des distributions de probabilités pour que B vérifie la condition de Markov.

De manière plus générale, les réseaux bayésiens (RB) sont des modèles graphiques pour représenter les relations probabilistes parmi un ensemble de variables aléatoires. Les RB offrent une représentation graphique de manière compacte des lois de probabilité jointes entre variables. La distribution de probabilités sur l'ensemble des variables est définie par :

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i/Pa(X_i)) \tag{1.3}$$

Où $Pa(X_i)$est l'ensemble des parents du nœud Xi dans G (**Pearl, 1985**).

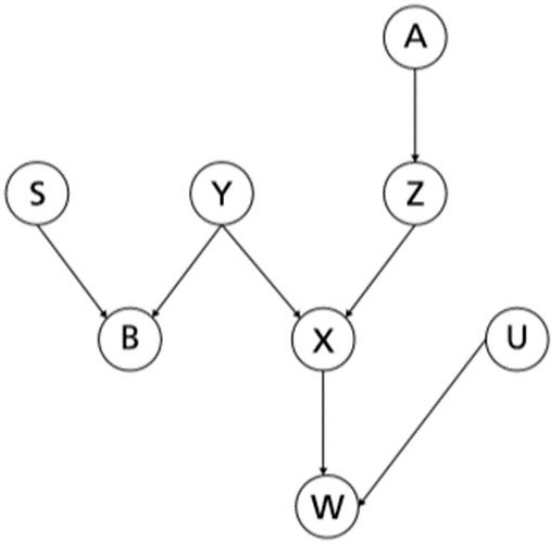


FIGURE 1.11 – Exemple de réseau bayésien

1.11 Conclusion

Dans ce chapitre, nous avons défini le concept d'analyse des big data. Par la suite, nous avons procédé à de brefs rappels sur les approches du data mining, leurs définitions, méthodes, applications dans le big data et leurs limites ainsi que des rappels sur les réseaux bayésiens.

CHAPITRE 2

Les réseaux bayésiens profonds

2.1 Introduction

Nous introduisons dans ce second chapitre la notion des réseaux bayésiens profonds, nous citons quelques travaux associés, nous effectuons par la suite une description de l'état de l'art des approches utilisées pour l'implémentation des réseaux bayésiens profonds et plus particulièrement l'approche de Monte Carlo Dropout.

2.2 Les réseaux bayésiens profonds

La principale motivation derrière la prise en compte des réseaux bayésiens profonds (Deep Bayesian Networks en anglais) est la nécessité de fournir une modélisation simplifiée et précise des données de grande dimension. Les défis des Big Data nécessitent un modèle robuste qui est appris à partir des données volumineuses capturées dans un petit intervalle de temps.

Il existe deux approches majeures pour l'apprentissage des réseaux bayésiens profonds. La première approche c'est d'apprendre une structure du réseau bayésien profond en proposant ses propres algorithmes de clustering (**Hasna Njah et al, 2016**). La deuxième approche est d'ajouter l'aspect bayésien aux architectures existantes du deep learning que nous appelons **l'apprentissage profond bayésien** (**Umara Zafar et al, 2019**).

2.3 Travaux associés

2.3.1 Architecture du réseau bayésien profond pour le Big data mining (Hasna Njah et al, 2016)

Dans cet article les auteurs se sont inspirés du principe de l'apprentissage et des réseaux bayésiens hiérarchiques afin de fournir une nouvelle architecture de réseau bayésien multi-couches avec des variables latentes. Cette architecture est appelée Deep Bayesian Network (Deep-BN). Il s'agit en effet d'un réseau bayésien arborescent où toutes les variables sont réparties en un nombre fini de couches ; la couche la plus basique est composée de toutes les caractéristiques observées. Ils ont proposé également une méthode triviale et nouvelle pour l'apprentissage de cette architecture. Ils présentent une étape de regroupement de caractéristiques qui vise à trouver les groupes de variables très dépendantes et une étape d'apprentissage de variables latentes qui trouve la distribution de probabilité des nouvelles variables cachées abstraites. Le nombre d'itérations de ces étapes, c'est-à-dire le nombre de couches cachées, dépend de la quantité de perte d'information entre deux couches successives. Par conséquent, leur architecture a l'avantage d'être simplement apprise et facilement interprétée en raison de l'aspect graphique du réseau bayésien. Celui ci permet de réduire la dimension des fonctions dans un contexte de méga données tout en conservant le plus d'information possible. Il offre également un rendement amélioré en matière de classification.

2.3.2 Réseau de neurones bayésien à convolution profonde pour l'analyse de la qualité des SMS (Yiping Du, 2019)

Pour obtenir une communication pratique et économique, le short message service (SMS) est l'un des moyens les plus faciles et les plus réalisables. Cependant, de plus en plus de messages indésirables tels que la publicité constituent un plus grand défi pour ce service. Dans cet article, les auteurs ont proposé une nouvelle architecture de réseau de neurones convolutifs "Convolutional neural network (CNN)" basée sur l'apprentissage bayésien pour réaliser une évaluation de la qualité des SMS. Plus précisément, ils forment d'abord le vecteur de mot de chaque mot dans l'ensemble de données du SMS, et convertissent le groupe de vecteurs de mots de chaque SMS en une matrice bidimensionnelle. Par la suite, la matrice de caractéristiques est utilisée comme entrée dans le réseau de neurones convolutifs, dans lequel les caractéristiques du SMS sont extraites par les noyaux de convolution, et l'extraction des caractéristiques est simplifiée par l'algorithme bayésien. Enfin, le score de qualité peut être atteint en fonction des caractéristiques optimales locales.

2.3.3 Construction des réseaux de neurones profonds par l'apprentissage de la structure d'un réseau bayésien (Raanan Y. Rohekar et al, 2018)

Il existe deux approches qui ont été étudiées pour l'apprentissage de la structure des modèles graphiques probabilistes. Plus précisément, les réseaux bayésiens pour l'estimation de la densité et la découverte des causes : L'une basée sur les scores et l'autre sur les contraintes. Motivés par les deux méthodes d'apprentissage de la structure des modèles graphiques probabilistes. Ces auteurs ont proposé une interprétation de la profondeur et de la connectivité inter-couches dans les réseaux de neurones profonds pour en extraire un Algorithme d'apprentissage de la structure de telle sorte qu'une hiérarchie des indépendances dans la distribution d'entrées est encodée dans un graphe génératif profond, où les indépendances d'ordre inférieur sont encodées dans des couches plus profondes. Ainsi, le nombre de couches est automatiquement déterminé, ce qui est souhaitable dans toute méthode d'apprentissage de l'architecture.

Les auteurs commencent par convertir le graphe génératif en un graphe discriminant, démontrant la capacité de ce dernier à imiter (préserver les dépendances conditionnelles) du premier. Dans la structure résultante, un neurone dans une couche est autorisé à se connecter aux neurones dans des couches plus profondes en sautant des couches intermédiaires. En outre, les neurones des couches plus profondes représentent des indépendances d'ordre faible (petits ensembles de conditions) et ont une large portée d'entrée, tandis que les neurones des premières couches représentent des indépendances d'ordre supérieur (ensembles de conditions plus grands) et ont une portée plus étroite.

2.3.4 Reconnaissance faciale avec les réseaux convolutifs bayésiens pour des systèmes de surveillance robustes (Umara Zafar et al, 2019)

La reconnaissance abstraite des images faciales est l'un des problèmes de recherche les plus difficiles dans les systèmes de surveillance en raison de différents problèmes, notamment la pose, l'expression, l'éclairage et la résolution. Le processus de reconnaissance faciale consiste à identifier la personne en comparant certaines caractéristiques d'une nouvelle personne (échantillon d'entrée) avec les personnes connues dans la base de données. La robustesse de la méthode de reconnaissance repose fortement sur la force des caractéristiques extraites et la capacité de traiter des images de visage de faible qualité. La capacité d'apprendre des caractéristiques robustes à partir d'images brutes du visage rend les réseaux neuronaux convolutionnels profonds (Dcnns) attrayants pour la reconnaissance du visage.

(**Umara Zafar et al, 2019**) ont utilisé la méthode dropout (le décrochage) pour former une architecture DCNN bayésien (B-DCNN). Il est montré qu'il y'a une relation entre le dropout et l'inférence variationnelle dans les B-DCNNs avec les distributions de Bernoulli

sur les poids du réseau. Dans ce travail, les auteurs ont utilisé cette approche pour représenter les incertitudes du modèle tout en classant les images faciales. Le décrochage est également utilisé au moment du test pour échantillonner la distribution postérieure sur les poids. La moyenne et la variance des échantillons sont utilisées respectivement comme confiance et incertitude pour chaque classe. La décision finale de classification est prise sur la base d'une fonction heuristique simple.

2.3.5 Réseaux de neurones bayésiens convolutifs avec inférence variationnelle approximative de Bernoulli (Gal et Ghahramani, 2015)

Les réseaux de neurones convolutifs (CNNs) fonctionnent bien sur de grands ensembles de données. Mais les données étiquetées sont difficiles à collecter. Donc le problème qui se pose est alors de savoir comment utiliser les CNNs avec de petits ensembles de données - car les CNNs tombent dans le surapprentissage rapidement. Dans cet article, les auteurs présentent un CNN bayésien efficace, offrant une résistance face au surapprentissage sur des petits ensembles de données contrairement aux anciennes approches. Pour ce faire, ils commencent par placer une distribution de probabilité sur les noyaux du CNN. Ils présentent de nouveaux résultats théoriques utilisant le dropout lors de l'apprentissage des réseaux autant qu'une inférence approximative dans les réseaux de neurones bayésiens. Cela a permet de réaliser le modèle employé dans cet œuvre en utilisant les outils existants sur le terrain sans augmentation de la complexité temporelle. Ils approximent la partie postérieure intraitable du modèle avec les distributions variationnelle de Bernoulli, sans avoir besoin de paramètres supplémentaires. Les résultats montrent une amélioration considérable de la précision de la classification par rapport aux techniques standards utilisées sur le CIFAR10.

2.3.6 Bayesian SegNet : Incertitude du modèle dans les architectures d'encodeurs-décodeurs convolutifs profonds pour la compréhension des scènes (Alex Kendall et al., 2017)

Dans cet article, ils présentent un cadre d'apprentissage profond pour la segmentation sémantique probabiliste en pixels, qu'ils appellent Bayesian SegNet. La segmentation sémantique est un outil important pour la compréhension de la scène visuelle et une mesure significative de l'incertitude est essentielle pour la prise de décision. Cette contribution est un système pratique qui est capable de prédire les étiquettes de classe pixel par pixel avec une mesure de l'incertitude du modèle en utilisant l'apprentissage bayésien profond. Ils parviennent par un échantillonnage Monte Carlo avec le dropout au moment du test pour générer une distribution postérieure des étiquettes des classes de pixels. Ils montrent aussi que l'incertitude de la modélisation améliore les performances de segmentation de 2 à 3 % pour un

certain nombre d'ensembles de données et d'architectures tels que SegNet, FCN, Dilation Network et DenseNet.

2.3.7 Le dropout comme approximation bayésienne : Représentation de l'incertitude du modèle dans l'apprentissage profond (Gal et Ghahramani, 2016)

Les outils d'apprentissage profond ont acquis une attention considérable dans les applications de l'apprentissage machine. Toutefois, ces outils de régression et de classification ne tiennent pas compte de l'incertitude du modèle. En comparaison, les modèles bayésiens offrent un cadre mathématiquement pour estimer l'incertitude du modèle, mais viennent généralement avec un coût de calcul énorme. Dans cet article, les auteurs développent un nouveau cadre théorique qui permet d'utiliser le dropout dans les réseaux de neurones profonds (NNs) comme inférence bayésienne approximative dans les processus gaussiens profonds. Un résultat direct de cette théorie, donne des outils pour modéliser l'incertitude avec les réseaux de neurones profonds en extrayant des informations des modèles existants qui ont été négligés jusqu'à présent. Cela permet de réduire le problème de la représentation de l'incertitude dans l'apprentissage profond sans sacrifier ni la complexité du calcul ni la précision des tests. Ils réalisent dans cet ?uvre, une étude approfondie des propriétés de l'incertitude du Dropout. Diverses architectures de réseau sont évaluées sur des tâches de régression et de classification, en utilisant le MNIST comme exemple. Ils approuvent une amélioration considérable de la fonction prédictive du log-vraisemblance ainsi que sur l'erreur quadratique moyenne par rapport aux méthodes existantes.

2.3.8 Génération de conversation émotionnelle basée sur un réseau neuronal bayésien profond (XIAO SUN et al, 2019)

Le domaine de la génération de conversations à l'aide de réseaux neuronaux attire de plus en plus l'attention des chercheurs depuis plusieurs années. Cependant, les modèles de langage neuronal traditionnels ont tendance à générer une réponse générique avec une mauvaise logique sémantique et aucune émotion. Cet article propose un modèle de génération de conversation émotionnelle basé sur un réseau neuronal bayésien profond qui peut générer des réponses avec des émotions riches, des thèmes clairs et des phrases diverses. Le sujet et les mots-clés émotionnels des réponses sont pré-générés en introduisant la connaissance du bon sens dans le modèle. La réponse est divisée en plusieurs clauses, puis un générateur multi-dimensionnel basé sur le mécanisme transformateur proposé dans cet article est utilisé pour générer itérativement des clauses à partir de deux dimensions : la granularité des phrases et la structure des phrases. Les expériences subjectives et objectives prouvent que, par rapport aux modèles existants, le modèle proposé améliore efficacement la logique sémantique et la précision émotionnelle des réponses. Ce modèle améliore également considérablement

la diversité des réponses, en surmontant largement les lacunes des modèles traditionnels qui génèrent des réponses sûres.

2.4 L'apprentissage profond bayésien

Le DEEP Learning a obtenu un succès significatif dans de nombreuses tâches de perception, y compris la reconnaissance visuelle des objets, la compréhension de textes et la reconnaissance vocale. Il s'agit sans aucun doute de tâches fondamentales pour un système complet d'intelligence artificielle (IA) ou d'ingénierie des données (DE) fonctionnel. Cependant, pour construire un véritable système d'IA/DE, le simple fait de pouvoir voir, lire et entendre est loin d'être suffisant. Il devrait surtout avoir la capacité de penser.

Dans presque tous les problèmes du monde réel, ce que nous voulons, ce n'est pas seulement un résultat, mais nous avons aussi besoin d'une connaissance de la confiance / certitude dans ce résultat. Lorsque l'apprentissage profond est utilisé dans des domaines sensibles tels que la santé et le contrôle autonome, nous devrions nous interroger non seulement sur la précision mais aussi sur la confiance des modèles déployés.

Prenons l'exemple du diagnostic médical. En plus de voir des symptômes visibles (ou des images médicales) et d'entendre des descriptions de patients, un médecin doit rechercher des relations entre tous les symptômes et de préférence inférer l'étiologie correspondante. Seulement après cela le médecin peut fournir des conseils médicaux pour les patients. Dans cet exemple, bien que les capacités de voir et d'entendre permettent au médecin d'acquérir de l'information auprès des patients, c'est la partie de la pensée qui définit un médecin. En particulier, la capacité de penser ici pourrait impliquer l'inférence causale, la déduction logique et le traitement de l'incertitude, ce qui est apparemment au-delà de la capacité des méthodes conventionnelles d'apprentissage en profondeur.

Un autre exemple, nous aimerions qu'un véhicule autonome non seulement identifie correctement les objets dans son environnement, mais fournisse également une mesure de l'incertitude. De même, si nous écrivons un bot qui négocie sur le marché de la bourse, nous voulons qu'il reconnaisse, quand la situation sort de sa zone de confort, de sorte qu'il peut arrêter d'agir et ne pas faire faillite.

Une grande partie de l'intelligence n'agit pas quand on est incertain. Il est donc surprenant que pour de nombreux projets de ML, exprimer l'incertitude n'est pas ce qui est visé.

Les outils standards d'apprentissage profond pour la régression et la classification ne saisissent pas l'incertitude du modèle. Dans la classification, les probabilités prédictives obtenues à la fin (la sortie softmax) sont souvent interprétées à tort comme une confiance du modèle (si c'est 0,8, alors mon modèle est sûr à 80 % de sa prédiction). Yarin Gal (**10**) argumente, qu'un modèle peut être incertain dans ses prédictions même avec une sortie softmax élevée.

Heureusement, un autre type de modèles existe, les modèles bayésiens, qui excellent dans l'inférence et la gestion de l'incertitude. Le problème est que le modèle bayésien n'est pas aussi bon que les modèles d'apprentissage profond pour les tâches de perception. Pour résoudre le problème, il est donc naturel d'intégrer étroitement l'apprentissage profond et le modèle bayésien dans un cadre probabiliste fondé sur des principes, que nous appelons l'apprentissage profond bayésien (BDL).

L'idée principale de l'apprentissage profond bayésien est d'introduire la notion de probabilité durant la phase d'entraînement et la phase de prédiction du réseau. Pour ce faire, les instances manipulées sont des distributions de probabilités et non des scalaires. Ainsi, les poids du réseau et les valeurs contenues des neurones suivent des lois normales qui sont caractérisées par une valeur moyenne et un écart-type. Ceci constitue la seule différence conceptuelle notable entre l'apprentissage profond et l'apprentissage profond bayésien (les autres différences étant d'ordre mathématique et algorithmique). Finalement, chaque neurone de la couche de sortie du réseau retourne une loi normale dont la valeur moyenne et l'écart type ont une valeur déterminée par l'image d'entrée du réseau. De ces distributions, il est possible à l'aide de formules mathématiques de calculer les différentes incertitudes (**11**).

Plusieurs formulations bayésiennes récentes du deep learning fournissent des techniques alternatives pour extraire des estimations d'incertitude à partir des réseaux profonds, y compris l'approche de Monte Carlo Dropout (**Gal et Ghahramani, 2016**) (qui sera développée par la suite), précédemment employée dans l'apprentissage actif profond pour la classification des images (**Gal et al, 2017**) et la reconnaissance des entités désignées (**Shen et al, 2018**) et l'approche de Bayes-by-Backprop. (**Blundell et al, 2015**).

2.5 Monte Carlo Dropout comme approximation bayésienne

2.5.1 Dropout

C'est une technique de régularisation (pour combattre l'overfitting) dont le principe est de désactiver aléatoirement à chaque itération un certain pourcentage des neurones d'une couche. Cela évite ainsi la sur-spécialisation d'un neurone (et donc l'apprentissage par cœur) (**Deshpande, 2016**). Il faut que la probabilité qu'un neurone ne soit pas désactivé soit entre 0.5 et 0.8 pour obtenir les meilleurs résultats (**Srivastava et al, 2014**). Ceci peut se faire par un tirage aléatoire. Par exemple, en jetant une pièce pour décider de la désactivation d'un neurone, on obtiendrait une probabilité de 0.5. Le nombre de neurones est donc réduit mais de façon aléatoire à chaque itération d'apprentissage. Cette technique fonctionne bien dans la généralisation d'un réseau.

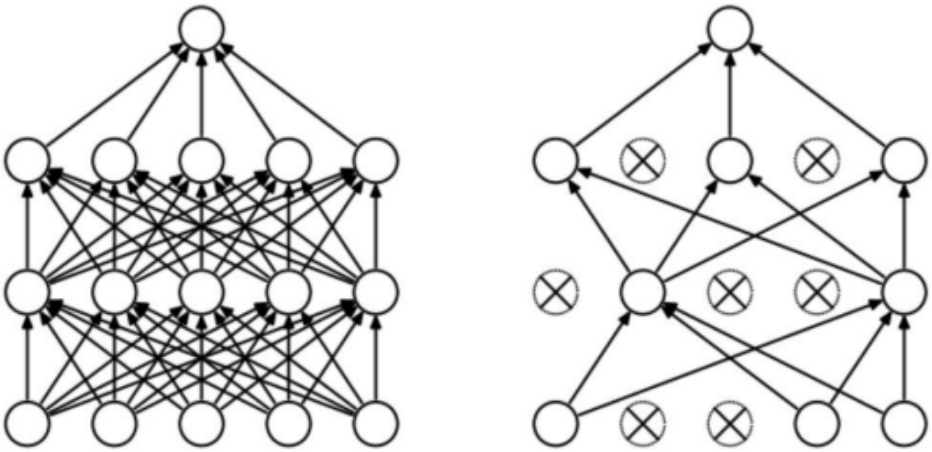


FIGURE 2.1 – Un réseau de neurones lors de l'application de la technique de Dropout (**Srivastava et al, 2014**)

Cette technique n'est utilisée que pendant la phase d'apprentissage. Le réseau neuronal se voit amputé d'une partie de ses neurones pendant la phase d'entrainement (leur valeur est estimée à 0) et ils sont par contre réactivés pendant la phase de test.

2.5.2 Estimation d'incertitude

La notion d'incertitude en apprentissage automatique se divise en deux catégories : les incertitudes épistémiques et les incertitudes aléatoires (**Der Kiureghian et Ditlevsen, 2009**). Bien que cette distinction entre incertitudes aléatoires et épistémiques semble abstraite au premier abord, elle représente, en réalité, des notions intuitives.

— Les incertitudes aléatoires sont associées à la qualité des données. Dans le cadre d'une base d'images, cela correspondrait à des incertitudes dûes au niveau de flou ou au manque de netteté de l'image.

— Les incertitudes épistémiques correspondent aux incertitudes dûes à la qualité du modèle. Il s'agit d'une incertitude liée au fait de ne pas connaître les meilleures valeurs de poids dans toutes les couches.

Ce sont ces dernières qui apportent le plus d'informations sur la validité de la prédiction du modèle (**Ronald Seoh, 2020**).

2.5.3 L'incertitude épistémique dans l'apprentissage profond bayésien (Ronald Seoh, 2020)

Compte tenu de l'ensemble de données $D(X, Y)$, où $X = x_1, ..., x_N$ contient les enregistrements de toutes les variables prédictives et $Y = y_1, ..., y_N$ porte la variable de résultat, nous aimerions prédire de nouveaux y^* étant donné un nouveau point de données w^*, avec un certain modèle de réseaux de neurones défini par l'ensemble de poids de couche

ou de paramètres W. Nous formulons généralement la tâche de trouver le W optimal comme problème d'optimisation et on le résout en utilisant des techniques telles que la descente de gradient stochastique. En conséquence, nous obtenons un seul ensemble W et donc une seule prédiction de y^*.

Par contre, dans les réseaux de neurones bayésiens on essaie de modéliser la distribution de W, et par la suite la distribution prédictive postérieure de y^*.

$$P(D) = \int P(D|W)P(W)d(W) \tag{2.1}$$

$$P(W|D) = \frac{P(D)P(W)}{P(D)} \tag{2.2}$$

$$P(y^*|x^*, D) = \int P(y^*|x^*, W)P(W|D)dW \tag{2.3}$$

Cependant, le problème ici est que le calcul de l'intégrale pour P(D) est faisable pour très peu de cas de réseaux de neurones or que ce n'est pas le cas pour le calcul de l'intégrale de la distribution postérieure de y.

Pour traiter des cas comme celui-ci où l'inférence exacte est difficile, les bayésiens ont développé deux familles majeures d'approches pour faire l'inférence approximative. L'une est l'inférence variationnelle (VI) et l'autre est Markov Chain Monte Carlo (MCMC). On s'intéresse dans ce travail à l'inférence variationnelle et plus particulièrement à l'une de ses variantes qui est le MC Dropout.

2.5.4 L'inférence variationnelle (VI) (Ronald Seoh, 2020)

La première idée clé dans l'inférence variationnelle est de remplacer $P(W|D)$ dans l'équation (2.3) par une distribution variationnelle approximative $q_\theta(W_i)$, qui est paramétrée par θ et peut être évaluée.

$$q_\theta^* = \int P(y^*|x^*, W)q_\theta(W)d(W) \approx P(y^*|x^*, D) \tag{2.4}$$

Nous voudrions que ce $q_\theta(W_i)$ ressemble le plus possible à $P(W|D)$. Idéalement, nous le ferions en minimisant la divergence Kullbeck-Leibler (KL) entre les deux :

$$KL(q_\theta(W)|P(W|D)) = \int q_\theta(W) \log \frac{q_\theta(W)}{P(D|W)}dW \tag{2.5}$$

Nous ne pouvons pas minimiser l'équation (2.5) car elle contient $P(W|D)$. Cependant, il a

été constaté que nous pouvons réduire au minimum l'équation (2.5) en maximisant la limite inférieure des données probantes (ELBO).

$$\psi_{v1}(\theta) = \int q_\theta(W) \log P(D|W) dW - KL(q_\theta(W)|P(W)) \leq \log P(D) \tag{2.6}$$

Il faut noter que le premier terme de l'équation (2.6) représente le log-vraisemblance conditionnelle.

2.5.5 L'approximation de Monte-Carlo Dropout

Cette méthode récemment inventée par (**Gal et Ghahramani, 2016**) s'appelle le **Monte-Carlo Dropout**. Il est intéressant d'expliciter les termes composant le nom de cette méthode : Monte-Carlo fait référence à une famille d'algorithmes effectuant des prédictions grâce à des processus aléatoires et le Dropout mentionné précédemment.

L'idée de cette technique est d'utiliser le dropout durant et la phase d'entrainement et la phase de test pour que par exemple à partir d'une seule entrée on puisse avoir plusieurs valeurs en sortie différentes à travers T MCD forward passes (quelques centaines de fois), c.à.d. passer la même entrée au réseau et par la suite appliquer un dropout aléatoire. Intuitivement, le modèle est capable de donner des prédictions différentes puisque différentes combinaisons de neurones sont utilisées pour la prédiction. En outre, cette méthode est en effet bayésienne.

De ces valeurs de sortie différentes, on peut obtenir une valeur moyenne et un écart-type par neurone de sortie. On obtient donc des résultats similaires aux résultats obtenus par un réseau bayésien ordinaire mais avec un réseau qui a été entraîné de manière classique (**11**) (**12**).

Dans MC dropout, nous définissons $q_\theta(W_i)$, la distribution approximative pour les poids de chaque couche i :

$$W_i = M_i.diag([Z_{i,j}]_{j=0}^{ki}) \tag{2.7}$$

$$Z_{i,j} \sim Bernoulli(p_i) pour i = 1, ..., L, j = 1, ..., K_{i-1} \tag{2.8}$$

Où chaque $Z_{i,j}$ est une variable aléatoire de Bernoulli décidant si l'entrée connectée doit être abandonnée ou non, avec une probabilité p_i. Par conséquent, p_i représente la probabilité de conserver l'entrée, ce qui est exactement le contraire de ce que nous appelons le taux de dropout (**Ronald Seoh, 2020**).

On souhaite calculer l'équation (2.6), mais il n'est pas possible d'évaluer directement la première partie avec intégrale. (**Gal et Ghahramani, 2016**) et (**Gal, 2016**) proposent une solution à ce propos qui peut se résumer en deux parties :

1. Nous pourrions rapprocher cette intégrale et obtenir un estimateur sans biais pour $\psi_{v1}(\theta)$, en minimisant les pertes typiques avec la régularisation L2, typiquement utilisé dans l'apprentissage des modèles de réseaux de neurones et prend la forme de :

$$\psi_{dropout} = \frac{1}{N}\sum_{i=1}^{N} E(y_i, y_i^{'}) + \lambda \sum_{i=1}^{L}(||W_i||_2^2 + ||b_i||_2^2) \tag{2.9}$$

   Où $E(y_i, y_i^{'})$ désigne des fonctions de perte telles que softmax ou l'erreur quadratique moyenne.

2. Cette inférence variationnelle est équivalente à une approximation par processus Gaussien (GP) du réseau neuronal avec une précision de modèle τ et une échelle de longueur l, ce qui signifie que nous obtenons une approximation de la distribution sur les fonctions possibles compte tenu des points de données.

   Nous pouvons maintenant obtenir un postérieur prédictif approximatif défini dans l'équation (2.3), ainsi que sa moyenne et sa variance :

$$P(y^*|x^*, D) = N(y^*; f^W(x^*), \tau^{-1}I) \tag{2.10}$$

$$E_{q\theta(y^*|x^*)}(y^*) \equiv \frac{1}{T}\sum_{t=1}^{T} f^W(x^*) \tag{2.11}$$

$$\begin{aligned} Var_{q\theta(y^*|x)}(y^*) \equiv \tau^{-1}I_D + \frac{1}{T}\sum_{t=1}^{T}(f^W(x^*))^T f^W(x^*) - \\ (E_{q\theta(y^*|x^*)}(y^*))^T E_{q\theta(y^*|x^*)}(y^*) \end{aligned} \tag{2.12}$$

   Où nous établissons la moyenne des T tests du réseau formé à l'équation (2.9).

   Il convient de noter que, bien que MC Dropout vise à saisir l'incertitude du modèle, sa formulation n'élimine pas complètement l'incertitude aléatoire. Au contraire, comme le montre l'équation (2.10), elle suppose une incertitude aléatoire, représentée par $\tau^{-1}I$, indiquant le bruit de données homogènes (**Ronald Seoh, 2020**).

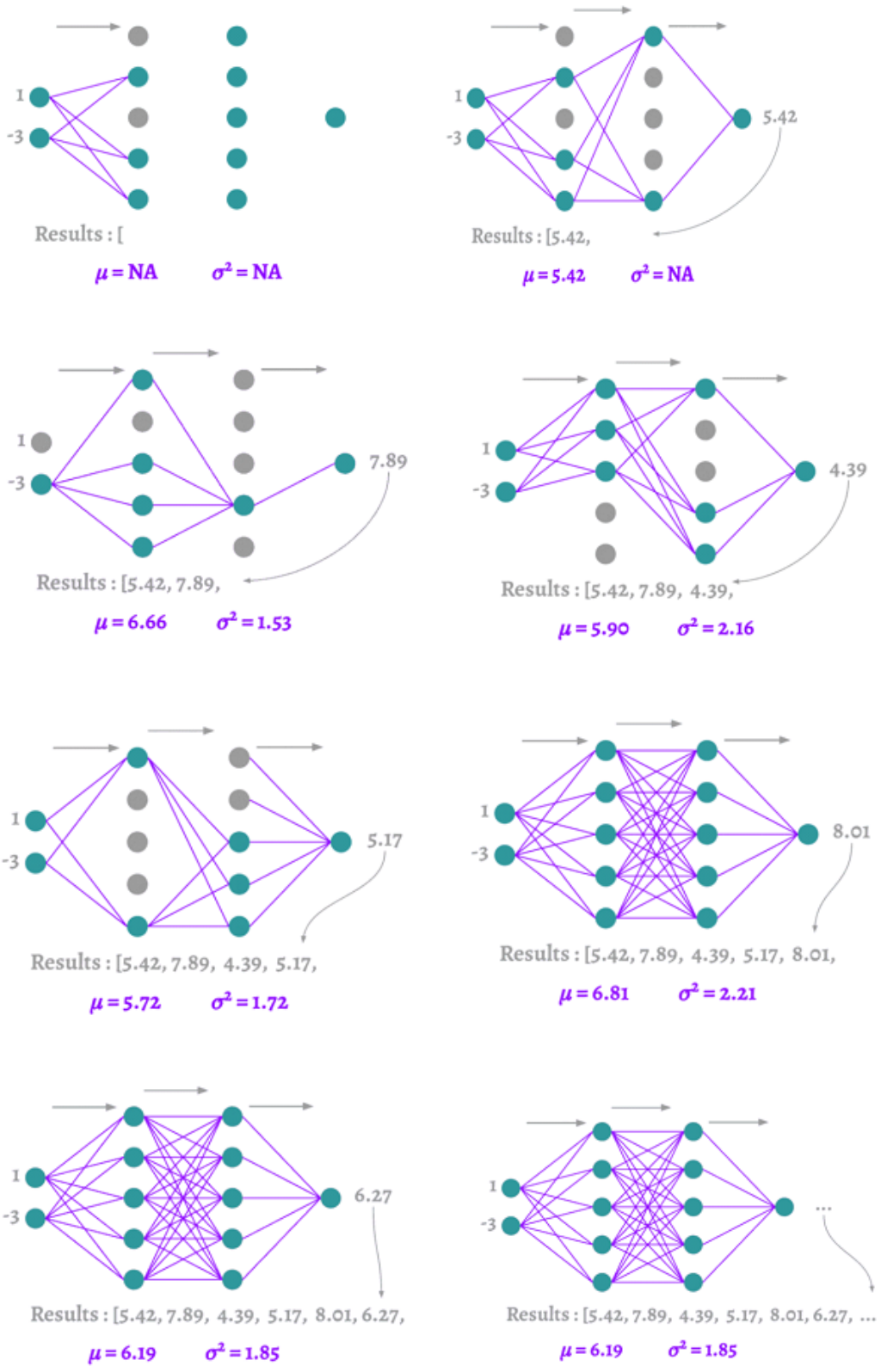


FIGURE 2.2 – Fonctionnement de dropout (**13**)

Ici, nous pouvons voir comment le dropout crée un ensemble de réseaux, qui donnent chacun une sortie différente, et cela nous permet d'avoir une distribution sur la sortie de notre réseau. De cette façon, nous pouvons voir qu'il y a des mesures d'incertitude, comme

la variance pour la régression ou l'entropie pour la classification.

Les avantages de cette méthode sont multiples en cela que l'entraînement du réseau ne nécessite aucun aménagement et peut être réalisé par le framework d'apprentissage profond de choix (Keras, Tensorflow, Pytorch, etc.). De plus, il est possible, en adaptant légèrement un réseau de neurones classique déjà entraîné, d'obtenir son approximation bayésienne, il suffit juste d'ajouter des couches de dropout après chaque couche avec des paramètres de poids, et de faire T prédictions de test. Le seul inconvénient de cette méthode est le temps de calcul lors de la prédiction (**11**).

2.6 Conclusion

Dans ce chapitre, nous avons présenté des généralités sur les réseaux bayésiens profonds, nous avons aussi expliqué le principe de l'approche de Monte Carlo Dropout. Une application de cette approche pour la classification dans un grand ensemble de données, fera l'objet du troisième chapitre.

CHAPITRE 3

Implémentation et résultats expérimentaux

3.1 Introduction

L'objectif de ce chapitre est de présenter les étapes de l'implémentation de l'approche de Monte Carlo Dropout. Nous allons donc nous intéresser au problématique de la classification d'images qui est la tâche d'attribuer à une image d'entrée x un label y à partir d'un ensemble fixe de catégories. C'est l'un des problèmes fondamentaux de la vision par ordinateur qui, malgré sa simplicité, a une grande variété d'applications pratiques. Pour cela, nous utilisons les réseaux de neurones convolutifs (CNN) qui sont les architectures état-de-l'art dans la quasi-totalité des tâches de classification liées aux images. Nous simulons le cas du big data en choisissant le big dataset CIFAR10.

Nous commençons tout d'abord par la présentation des ressources, du langage et de l'environnement de développement que nous avons utilisé. Puis les étapes de la réalisation du modèle et on termine par les tests effectués.

Ce chapitre est composé de deux parties, l'implémentation du système et les résultats expérimentaux des tests.

3.2 Environnement et outils de travail

Nous allons présenter les différents logiciels, langages et libraires utilisés pour implémenter notre approche proposée.

3.2.1 Environnement Matériel

Le matériel utilisé le long de ce projet consiste en deux ordinateurs personnels ainsi qu'un serveur (Cloud). Nous nous sommes tournées vers les serveurs fournis par l'outil Google Colab. Etant donné le temps considérable que prend l'opération d'apprentissage des modèles sur nos propres machines.

Poste de travail 1 :

Processeur	Intel(R)® Core i3-4005U CPU @ 1.70 GHz
RAM	4.00 Go

TABLE 3.1 – Caractéristiques du poste de travail 1

Poste de travail 2 :

Processeur	Intel(R)® Core (TM) i5-3230M CPU @ 2.60 GHz
RAM	4.00 Go

TABLE 3.2 – Caractéristiques du poste de travail 2

Serveur Cloud (Google Colaboratory) :

Processeur	(2x) Intel(R) Xeon(R) CPU @ 2.20GHz
RAM	13 Go
Processeur graphique	Tesla K80

TABLE 3.3 – Caractéristiques du service Colab (Google Colaboratory)

3.2.2 Langages de programmation et logiciels

Nous avons utilisé au cours de la réalisation de notre système plusieurs langages de programmation, bibliothèques, outils et logiciels. Voici une brève présentation de chacun de ces derniers :

Python

Python est un langage de programmation de haut niveau. Il supporte la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort et d'une gestion automatique de la mémoire. Il est réputé pour être un langage simple à utiliser. Plusieurs bibliothèques sont fournies afin de faciliter les développements (**14**).

Anaconda

Anaconda est une distribution libre et open source3 de Python appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique. Les versions de paquetages sont gérées par le système de gestion de paquets conda5. L'avantage de ces distributions est de pouvoir installer plus facilement les librairies sans soucis de compatibilité entre différents paquets.La distribution Anaconda est utilisée par plus de 20 millions d'utilisateurs et comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOS (**15**).

FIGURE 3.1 – Logo du langage de programmation Python et de la distribution Anaconda

3.2.2.1 Librairies et bibliothèques

Pytorch

PyTorch est une bibliothèque logicielle Python d'apprentissage automatique qui s'appuie sur Torch développée par Facebook. Elle permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond. Ces calculs sont optimisés et effectués soit par le processeur (CPU) soit, lorsque c'est possible, par un processeur graphique (GPU) supportant CUDA. Il est issu des équipes de recherche de Facebook, et avant cela de Ronan Collobert dans l'équipe de Samy Bengio à l'IDIAP (**16**).

Numpy

Pour Numerical Python, est une bibliothèque qui permet d'effectuer des calculs numériques avec le langage Python. Elle introduit une gestion facilitée des tableaux de nombres, qui sont d'une certaine manière, comme les listes en Python, mais Numpy permet de rendre la manipulation des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux,beaucoup plus efficaces, surtout sur les tableaux de large taille. Les tableaux Numpy sont au cœur de presque tout l'écosystème de data science en Python (**17**).

Matplotlib

Pour Mathematic Plot library est une bibliothèque gratuite complète pour créer des visualisations statiques, animées et interactives en Python (**18**).

Html

Le HyperText Markup Language, généralement abrégé HTML ou dans sa dernière version HTML5, est le langage de balisage conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement la page, de mettre en forme le contenu, de créer des formulaires de saisie, d'inclure des ressources multimédias dont des images, des vidéos, et des programmes informatiques (**19**).

Css

Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000 (**20**).

JavaScript

JavaScript JS est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les technologies HTML et CSS (**21**).

Eel

Eel est une petite bibliothèque Python pour créer des applications GUI HTML/JS de type électronique. Ceci utilisé pour créer des interfaces graphiques dans une fenêtre d'application Chrome avec HTML, CSS et JS. En résumé, il héberge un serveur web local, puis fournit des fonctionnalités pour communiquer entre JavaScript et Python (**22**).

Figure 3.2 – Logos de quelques librairies utilisées

3.2.2.2 Logiciels et éditeurs de texte

Pycharm

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django.

Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache (**23**).

Jupyter Notebook

Jupyter est une interface web dans laquelle il est possible d'utiliser et d'éditer du code en Python (ainsi que plusieurs autres langages), de l'exécuter et de voir directement les résultats, comprenant également une visualisation à l'aide de graphiques (**24**).

FIGURE 3.3 – Logos des logiciels utilisés

3.2.2.3 Gestion de version

Git

Un logiciel libre de gestion de version. Il se distingue par sa rapidité et sa gestion des branches qui permettent de développer en parallèle de nouvelles fonctionnalités.

FIGURE 3.4 – Logo de Git

3.2.3 Description du dataset

Le CIFAR-10 est un sous-ensemble étiqueté parmi 80 millions de jeux de données d'images. Il a été recueilli par Alex Krizhevsky, Vinod Nair, et Geoffrey Hinton.

La base d'image de CIFAR-10 se compose de 60000 images couleur, chaque image à une taille de 32x32, ces images sont réparties en 10 classes, avec 6000 images par classe. Dans cette base on trouve 50000 images pour l'apprentissage et 10000 images pour le test (**25**).

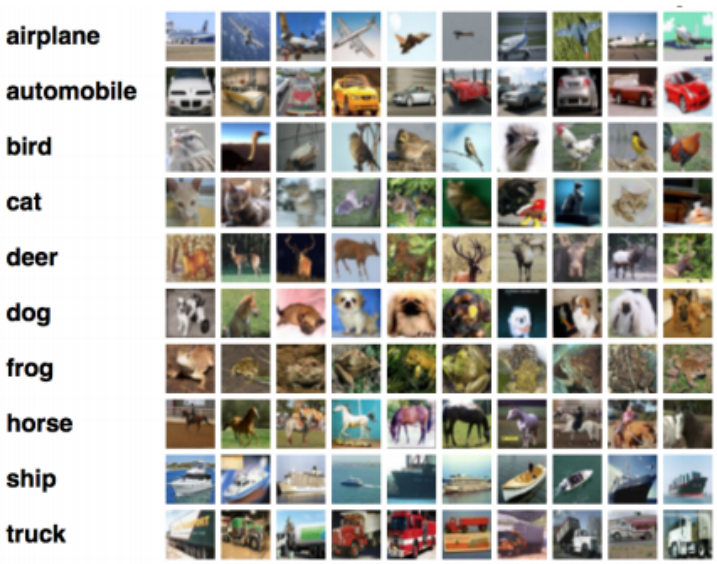


FIGURE 3.5 – Exemples de 10 images aléatoires de chacune des 10 classes (**25**)

3.2.3.1 Préparation des données

Une technique courante en machine learning est de normaliser les données afin de mieux conditionner l'apprentissage. En apprentissage avec les CNN, la technique la plus courante est de calculer sur l'ensemble d'entrainement la valeur moyenne μ et l'écart type σ de chaque canal RGB. On obtient donc 6 valeurs. On normalise ensuite chaque image en soustrayant à chaque pixel la valeur moyenne correspondant à son canal et en la divisant par l'écart type correspondant.

Pour CIFAR-10, les valeurs sont $\mu = [0.491, 0.482, 0.447]$ et $\sigma = [0.202, 0.199, 0.201]$. Mais, avant de réaliser l'étape de normalisation on doit tout d'abord convertir nos images en tenseurs pour pouvoir les manipuler facilement par la suite.

Le réseau de neurone convolutif nécessite de grands ensembles d'images d'entrainement pour obtenir un bon résultat. Les données disponibles pour l'entrainement sont divisées en deux ensembles différents : Ensemble d'apprentissage et Ensemble de validation. Il ne devrait pas y avoir de chevauchement entre ces deux ensembles des données afin d'améliorer la capacité de généralisation du réseau de neurones. Les performances réelles d'un réseau ne sont révélées

que lorsque le réseau est testé avec des données de test pour mesurer le rendement du modèle sur les données qui n'ont pas été vues pendant l'apprentissage. Le test est conçu pour accéder à la capacité de généralisation du réseau. Une bonne généralisation signifie que le réseau fonctionne correctement sur des données similaires, mais différentes des données d'apprentissage.

3.3 Architecture et description

3.3.1 Architecture proposée

Les réseaux de neurones convolutifs (CNN) sont devenus les architectures état-de-l'art dans la quasi-totalité des tâches de machine learning liées aux images.

Le modèle que nous proposons est composé de deux couches de convolution, deux couches de maxpooling ainsi que trois couches entièrement connectées, l'architecture du modèle se présente comme suit :

Input > Conv (ReLU) > MaxPool > Conv (ReLU) > MaxPool > FC (ReLU) > FC (ReLU) > FC > 10 outputs

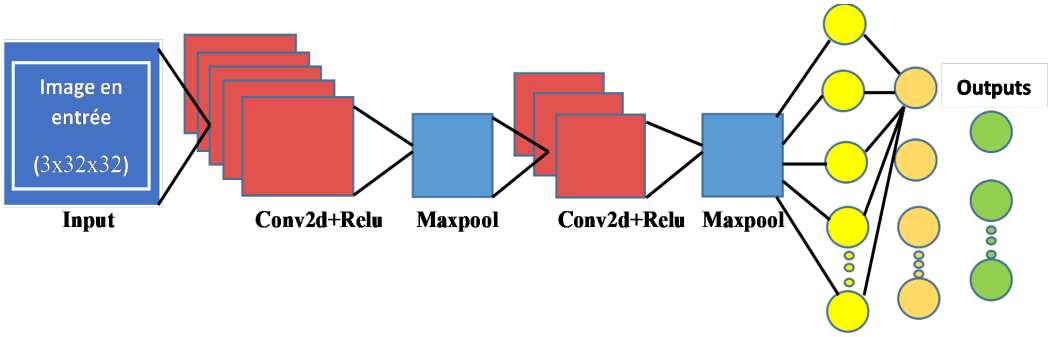


FIGURE 3.6 – Architecture du modèle proposé

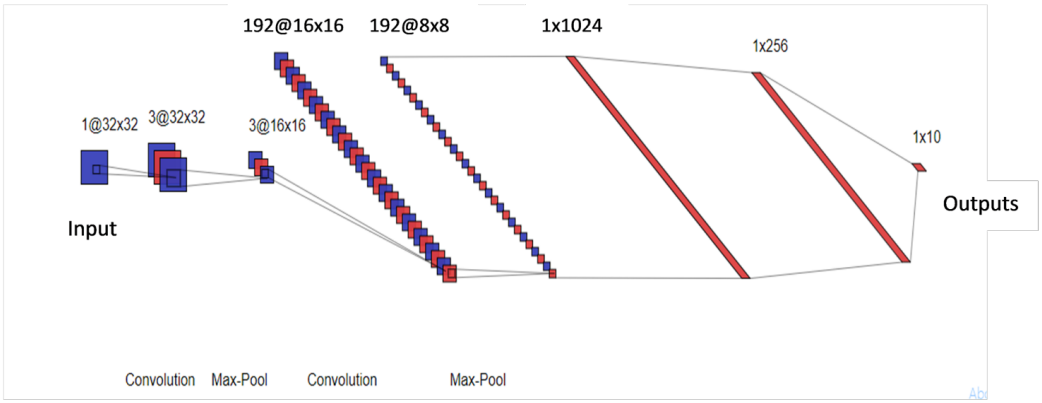


FIGURE 3.7 – Architecture du réseau utilisé faite à l'aide de l'outil : **alexlenail.me**

Les images en entrées sont de format 3x32x32, .c.à.d. 3 canaux (RGB) chacune de taille 32x32 pixels.

L'image passe d'abord par la première couche de convolution, cette couche est composée de 192 filtres chacun de taille 5x5, après cette convolution 192 feature maps de taille 32x32 seront créés.

Par la suite, une couche de maxpooling est appliquée à ces feature maps avec un kernel de 2x2 qui va réduire la taille de la sortie à 192x16x16, et donc on aura 192 feature maps de taille 16x16.

La 2eme couche de convolution va utiliser 192 autres filtres de taille 5x5 pour appliquer une convolution aux 192 feature maps en sortie de la couche précédente.

Une dernière couche de maxpooling est appliquée pour réduire la taille des feature maps en sortie de 192x16x16 à 192x8x8.

La sortie de la couche finale du maxpooling doit être aplatie afin que nous puissions la connecter à une couche entièrement connectée. Pour ce faire, on utilise la méthode du torch.tensor.view, en spécifiant -1, la méthode déduira automatiquement le nombre de lignes adaptés aux nombre de colonnes en entrées. Ceci est fait pour traiter la taille des mini batch de données.

La 1ere couche entièrement connectée utilise une fonction d'activation Relu et elle est composée de 1024 neurones.

La 2ème couche entièrement connectée utilise également une fonction d'activation Relu et est connectée à la couche précédente avec 256 neurones.

Enfin, La 3ème couche entièrement connectées est liée aux 256 sorties de la couche précédente pour pouvoir à la fin, générer 10 outputs (une pour chaque classe du CIFAR-10), Notons que nous n'avons pas utilisé une activation dans cette couche en raison de l'utilisation de la fonction CrossEntropyLoss qui combine à la fois une activation SoftMax et une fonction de perte cross entropy.

3.3.2 Application du dropout

Nous avons proposé deux modèles où la différence se situe dans les couches de l'application du dropout. Nous allons comparer les performances des deux méthodes sur la base des résultats des expérimentations.

— **1er modèle (Convolutional and fully-connected dropout)**
Nous avons appliqué le dropout sur les couches de convolutions (conv1, conv2) ainsi que sur les couches entièrement connectées (fc1, fc2).

— **2ème modèle (Max-pooling and fully-connected dropout)**
Nous avons appliqué le dropout sur les deux couches maxpooling ainsi que sur les

couches entièrement connectées (fc1, fc2).

3.3.3 Fonction de perte et algorithme d'optimisation

Nous avons choisi la fonction de perte CrossEntropyLoss car elle est convenable pour les problèmes de classification en k classes $k \geq 3$, et aussi car elle minimise la distance entre deux distributions de probabilité (les valeurs prévues et les valeurs réelles). Concernant l'optimiseur (algorithme d'optimisation), nous avons choisi d'utiliser les méthodes SGD, Adam et Adadelta.

3.3.4 Apprentissage du réseau

Nous allons maintenant entrainer le réseau en utilisant les données du trainloader, en parcourant toutes les données d'entrainement par batch de 4 images, et en répétant l'ensemble du processus autant de fois que nécessaire pour ne pas tomber dans le surapprentissage. Après chaque 2000 batch, nous affichons quelques statistiques concernant le progrès de l'apprentissage : l'époque actuelle, l'étape courante ainsi que la valeur de la fonction de perte.

Nous allons tester en parallèle l'efficacité de notre modèle sur un ensemble de validation en calculant la précision du modèle (accuracy) ainsi que la valeur de la fonction de perte (Loss) pour pouvoir par la suite détecter le surapprentissage et après cela choisir le nombre d'époques optimal pour notre apprentissage.

A la fin de chaque époque, nous affichons la valeur de précision et la valeur de perte de l'ensemble d'apprentissage ainsi que celles de l'ensemble de validation.

3.3.5 MC-Dropout Test

Cette partie représente le cœur de notre approche où nous allons introduire l'aspect bayésien. Contrairement aux utilisations standard du dropout, qui se suffit d'utiliser le dropout durant la phase d'apprentissage du réseau, nous allons étendre son utilisation pour le test aussi (seulement pour les couches du dropout). Par la suite, nous générons pour chaque entrée une liste de prédictions à travers plusieurs MCD forward passes. Nous calculons ensuite la moyenne des prédictions sur les T itérations qui va être utilisée comme moyenne finale des prédictions sur l'échantillon de test. La classe avec la moyenne prédictive la plus élevée est sélectionnée comme prédiction finale de la sortie. D'autre part, nous allons utiliser les listes de prédictions à travers plusieurs forward passes pour calculer l'incertitude de chaque classe. Nous commençons par générer une moyenne des prédictions pour chaque classe tout au long des T itérations, cette moyenne est calculée par lot et ensuite pour tout l'ensemble de test, et enfin, nous allons utiliser cette moyenne pour mesurer l'entropie de chaque classe qui va nous permettre de déduire l'incertitude du modèle pour chaque classe.

3.4 Expérimentation, évaluation et discussion des résultats

Dans cette parie, nous exposons les différentes expérimentations menées ainsi qu'une étude comparative des deux modèles proposés précédemment. Nous illustrons les résultats en termes de précision et d'erreur pour aboutir à la fin à un résultat probant.

➤ **Nombre d'époques (epochs)**

Nous avons effectué un premier apprentissage avec 10 époques mais celui-ci progressant encore, nous l'avons rechargé et augmenté à 75 jusqu'à 100 itérations. Après plusieurs tests et exécutions successifs, nous avons vite remarqué que la plupart de nos apprentissages se stabilisent entre 20 et 30 itérations, valeurs que nous avons gardées tout au long de nos expérimentations, sauf pour l'optimiseur SGD qui parfois continue à progresser tout au long de 100 itérations (figure 3.7).

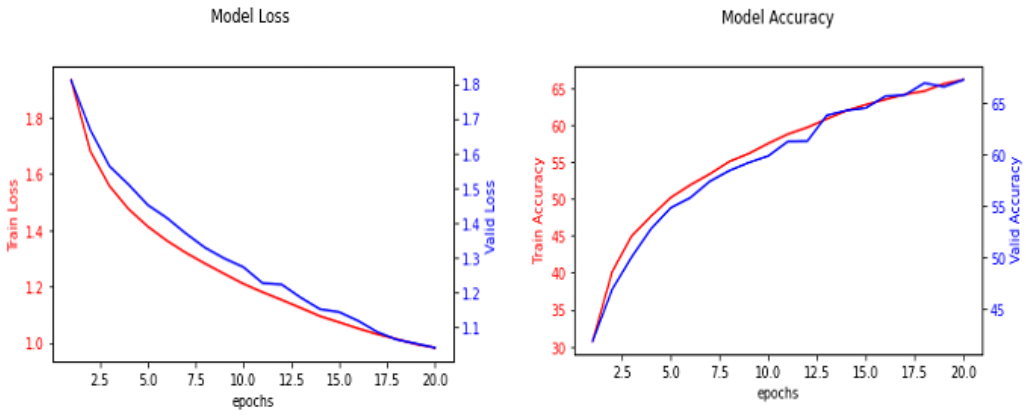


FIGURE 3.8 – Courbes de précision et de perte du 2eme modèle avec **SGD**, $lr = 0.0001$

➤ **learning rate**

Nous avons exécuté successivement des apprentissages avec plusieurs valeurs de learning rate, allant de 0.0001 jusqu'à 0.01, avec les différents algorithmes d'optimisation définis précédemment.

3.4.1 Résultats de l'étape d'apprentissage

3.4.1.1 Résultats obtenus pour le 1 er modèle

➤ **L'optimiseur Adadelta**

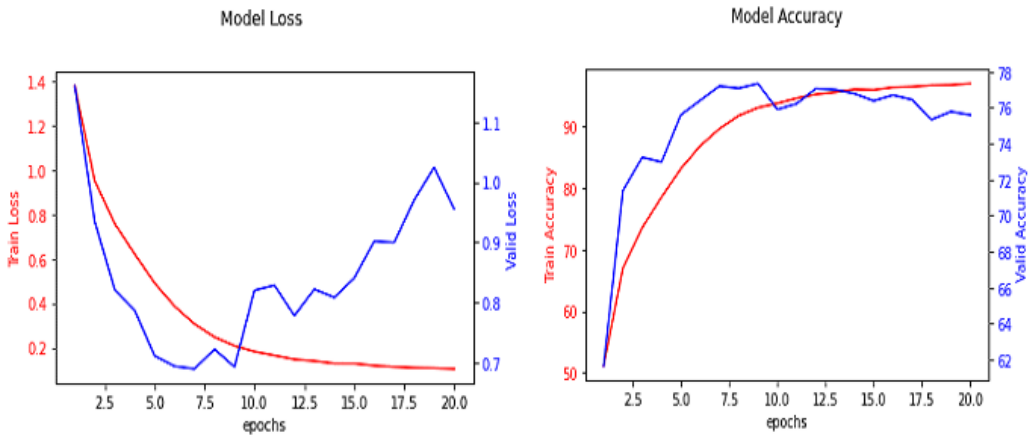


FIGURE 3.9 – Courbes de précision et de perte du 1 er modèle avec **Adadelta**, $lr = 0.001$

➤ **L'optimiseur Adam**

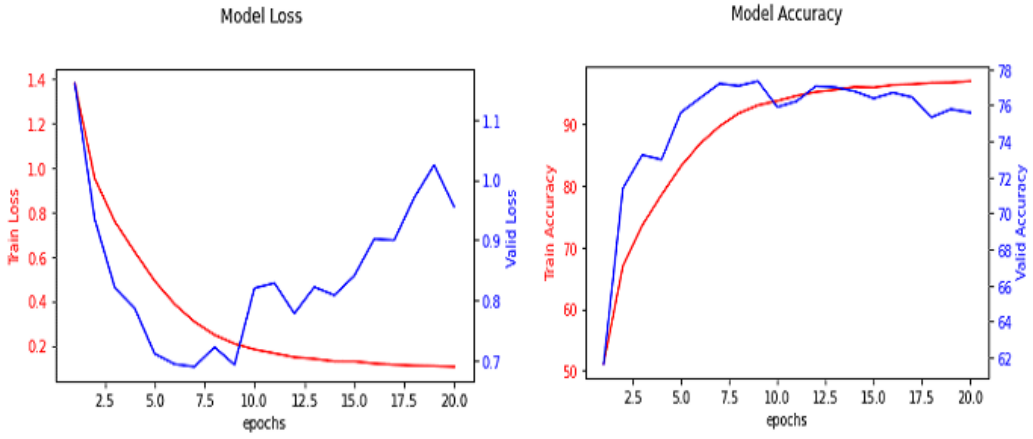


FIGURE 3.10 – Courbes de précision et de perte du 1 er modèle avec **Adam**, $lr = 0.0001$

➤ **L'optimiseur SGD**

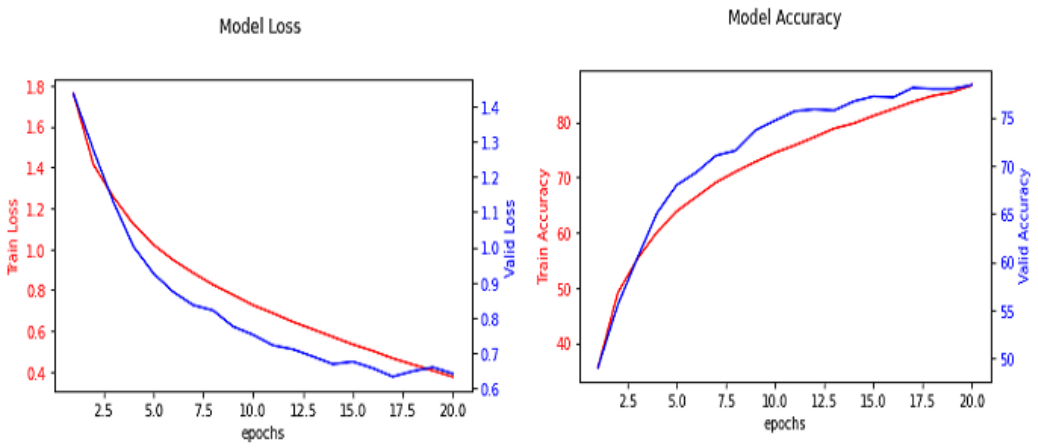


FIGURE 3.11 – Courbes de précision et de perte du 1 er modèle avec **SGD**, $lr = 0.001$

Discussion des résultats

Les tests effectués avec Adam sur le modèle 1 n'ont pas été satisfaisants, malgré qu'ils atteignent une précision de 77% avec un $lr = 0.\ 0001$. Cependant le modèle tombe rapidement dans le surapprentissage après seulement 7 epochs, et ne dépasse pas les 10% pour les autres valeurs du learning rate.

Avec une valeur de $lr = 0.01$, Adadelta a atteint une valeur de précision de 74%. Diminuer la valeur du learning rate à 0.001 puis à 0.0001 engendre les précisions de 67% et 50%.

Quant à SGD, c'est la méthode donnant le résultat le plus prometteur mais qui reste néanmoins insuffisant.

Optimiseur	Précision Max
SGD	78%
Adadelta	67%
Adam	77%

TABLE 3.4 – Les résultats obtenus par les différents optimiseurs utilisés dans le premier modèle

3.4.1.2 Résultats obtenus pour le 2ème modèle

➤ **L'optimiseur Adadelta**

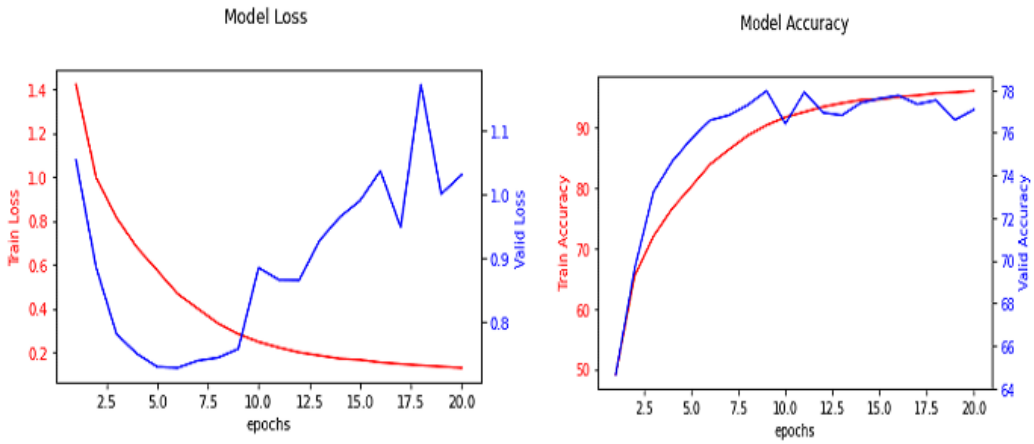


FIGURE 3.12 – Courbes de précision et de perte du 2ème modèle avec **Adadelta**, $lr = 0.01$

➤ **L'optimiseur Adam**

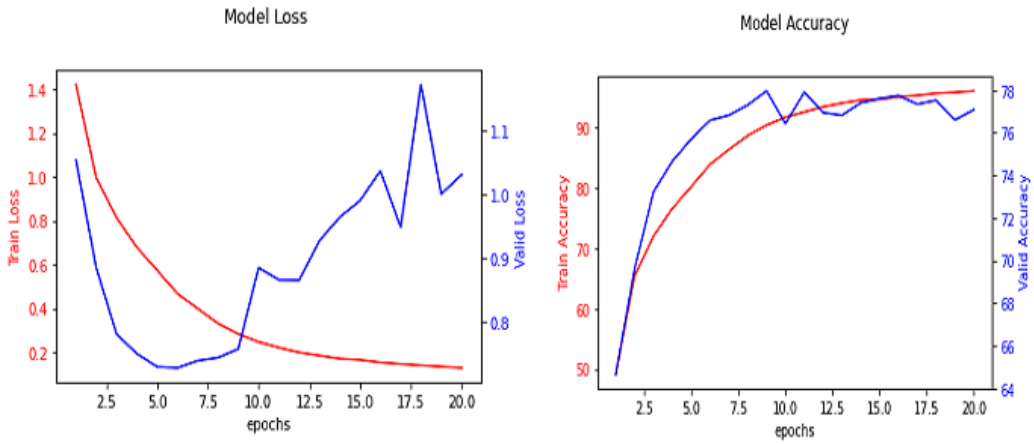


FIGURE 3.13 – Courbes de précision et de perte du 2ème modèle avec **Adam**, $lr = 0.0001$

➤ **L'optimiseur SGD**

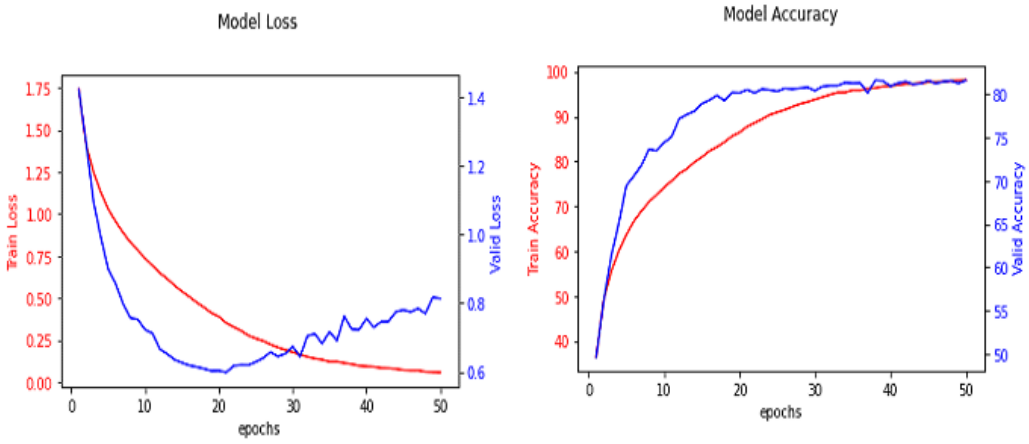


FIGURE 3.14 – Courbes de précision et de perte du 2ème modèle avec **SGD**, $lr = 0.001$

Discussion des résultats

Les tests effectués sur le 2eme modèle ont montré que les meilleurs résultats sont obtenus en utilisant l'optimiseur SGD qui atteint 82% avec un $lr = 0.\ 001$. Néanmoins, les précisions diminuent en baissant les valeurs de learning rate (78% avec $lr = 0.001$ et 73% avec $lr = 0.0001$).

Malgré sa popularité, les tests effectués avec Adam sur le 2eme modèle n'ont pas été satisfaisants. Bien qu'ils atteignent une précision de 76% avec un $lr = 0.0001$, l'apprentissage avec Adam a mené vers un surapprentissage assez rapidement, ne dépassant pas les dix itérations, ainsi qu'une valeur de précision qui ne dépasse pas les 10% avec différentes valeurs de learning rate. c En ce qui concerne l'optimiseur Adadelta, le modèle a atteint des valeurs de précisions entre 74%,72%,42% en diminuant la valeur du learning rate respectivement de 0.01, 0.001, 0.0001.

L'utilisation prometteuse du SGD pourrait être améliorée en augmentant le nombre de données, ce qui implique plus de calculs et donc requiert plus de ressources.

Optimiseur	Précision Max
SGD	82%
Adadelta	74%
Adam	76%

TABLE 3.5 – Les résultats obtenus par les différents optimiseurs utilisés dans le deuxième modèle

3.4.2 Conclusion

D'après les différentes expérimentations et configurations testées, le meilleur taux de précision a été obtenu avec l'architecture du modèle 2 **(Max-pooling and fully-connected dropout)**, avec application du dropout sur les couches de pooling et des couches entièrement connectées, en utilisant l'optimiseur SGD avec un $lr = 0.001$ Dans l'ensemble, cette dernière a globalement surpassé les performances obtenues avec le modèle1.Ce résultat nous a permis d'utiliser le modèle 2 **(Max-pooling and fully-connected dropout)** dans le MC Dropout Test.

3.4.3 Résultats du MC Dropout test

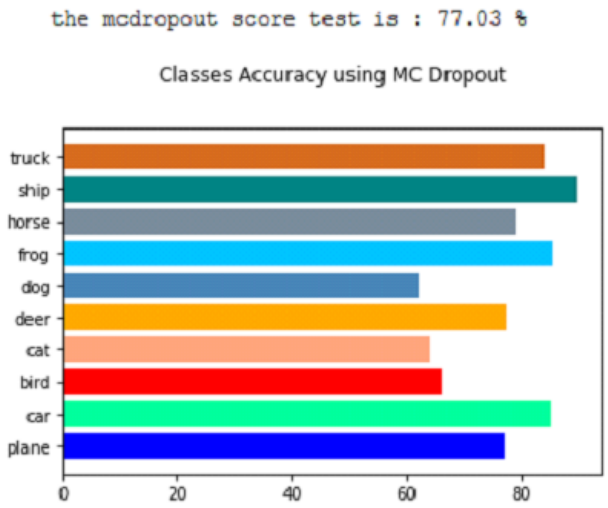


FIGURE 3.15 – Score du MC Dropout test et précisions de chaque classe

On peut voir que le modèle a atteint une précision moyenne de 77% sur les données de test.

On peut aussi visualiser les résultats de précision de chaque classe qui varient entre un minimum de 63% pour la classe « dog » et un maximum de 95% pour la classe « ship ».

3.4.4 Résultats d'incertitude du modèle

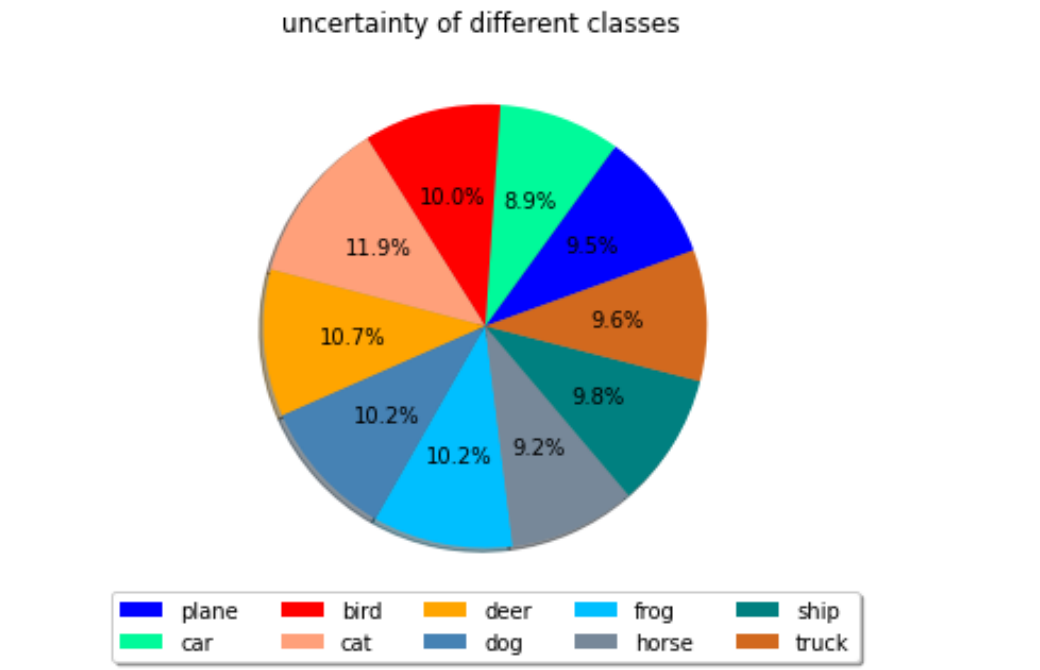


FIGURE 3.16 – Pourcentage d'incertitude des différentes classes

On peut bien voir que l'incertitude de chaque classe ne dépasse pas les 11.9%, et possède un taux minimal de 8.9%.

Ces résultats peuvent être très utiles lors de la prise de décision car on peut améliorer les résultats d'incertitude d'une classe donnée en prévoyant par exemple plus de données en entrée pour celle-ci pour permettre au modèle de faire de bonnes prédictions, chose qui va nous permettre d'aboutir à un gain de temps énorme, élément essentiel dans tout problème de prise de décision.

3.5 Réalisation de l'application

En raison d'absence du matériel assez puissant, nous avons réduit quelques paramètres de notre modèle afin d'accélérer le processus.

Nous avons utilisé deux couches de convolution composées de 6 filtres chacun de taille 5x5, deux couches de max pooling de taille 2x2, trois couches entièrement connectées : La 1ere couche entièrement connectée utilise une fonction d'activation Relu et elle est composée de 120 neurones, la 2 ème utilise également une fonction d'activation Relu et est connectée à la couche précédente avec 84 neurones et la 3 ème couche connectées est liée aux 84 sorties de la couche précédente pour pouvoir à la fin, générer 10 outputs.

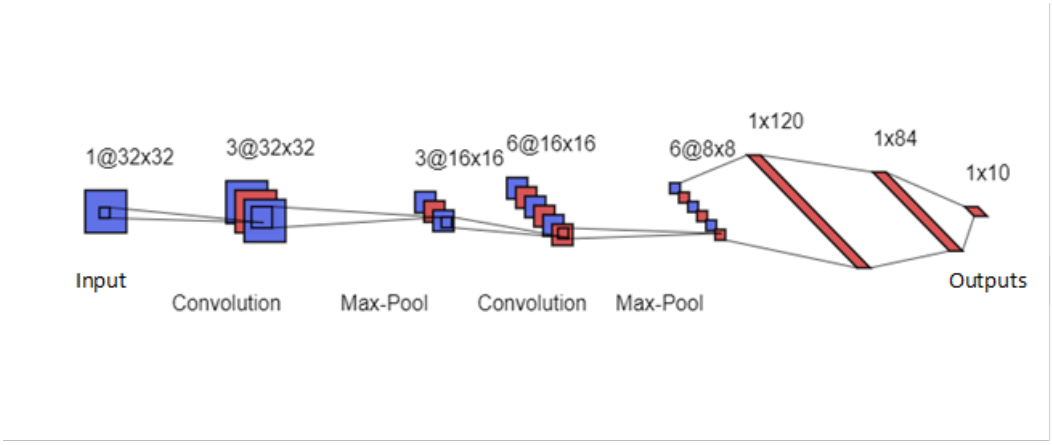


FIGURE 3.17 – Architecture du réseau utilisé faite à l'aide de l'outil : **alexlenail.me**

Nous avons tenu à introduire notre réseau de neurones profond bayésien dans une interface graphique afin de pouvoir visualiser les performances facilement sans être obligé de passer à chaque fois par les lignes de codes. Les détails de l'interface sont montrés dans la figure suivante :

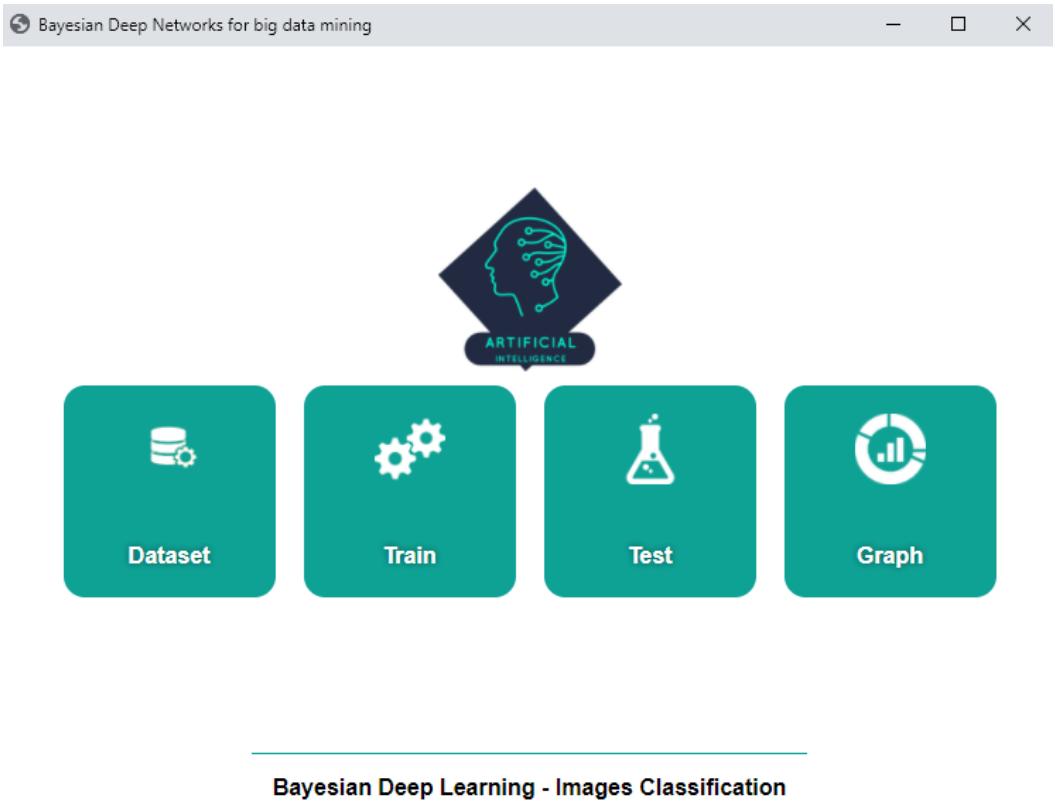


FIGURE 3.18 – L'interface de l'application

➤ Dataset : un petit aperçu sur la data set utilisée. Il est très important de visualiser nos données avant l'entrainement. Elles peuvent donner un aperçu des raisons pour lesquelles le modèle ne se comporte pas comme prévu. Il peut y avoir des situations où les données du modèle appartiennent entièrement ou principalement à une classe, c'est-à-dire des situations où le modèle est biaisé. Cela est principalement dû à un ensemble de données déséquilibré. Ainsi qu'au problème de manque de données pour renforcer l'énoncé du problème (**26**).

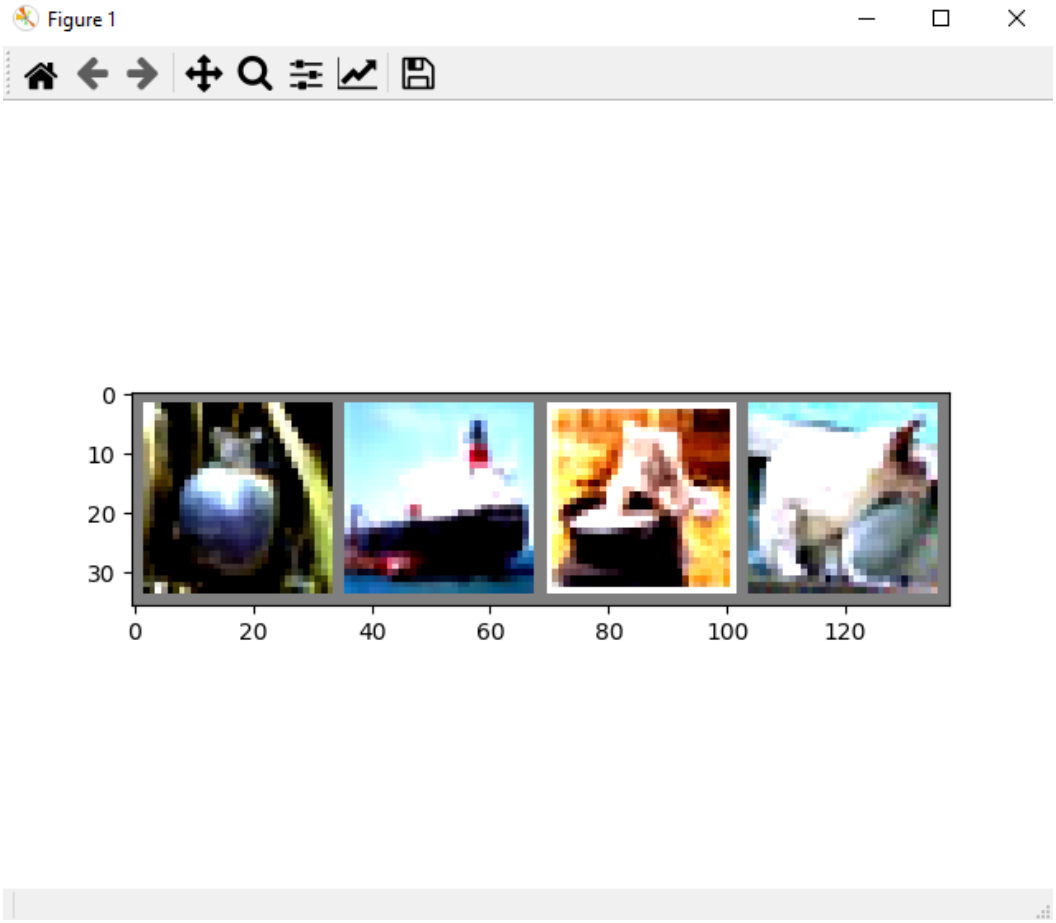


FIGURE 3.19 – Affichage d'un ensemble aléatoire des images d'entrainement

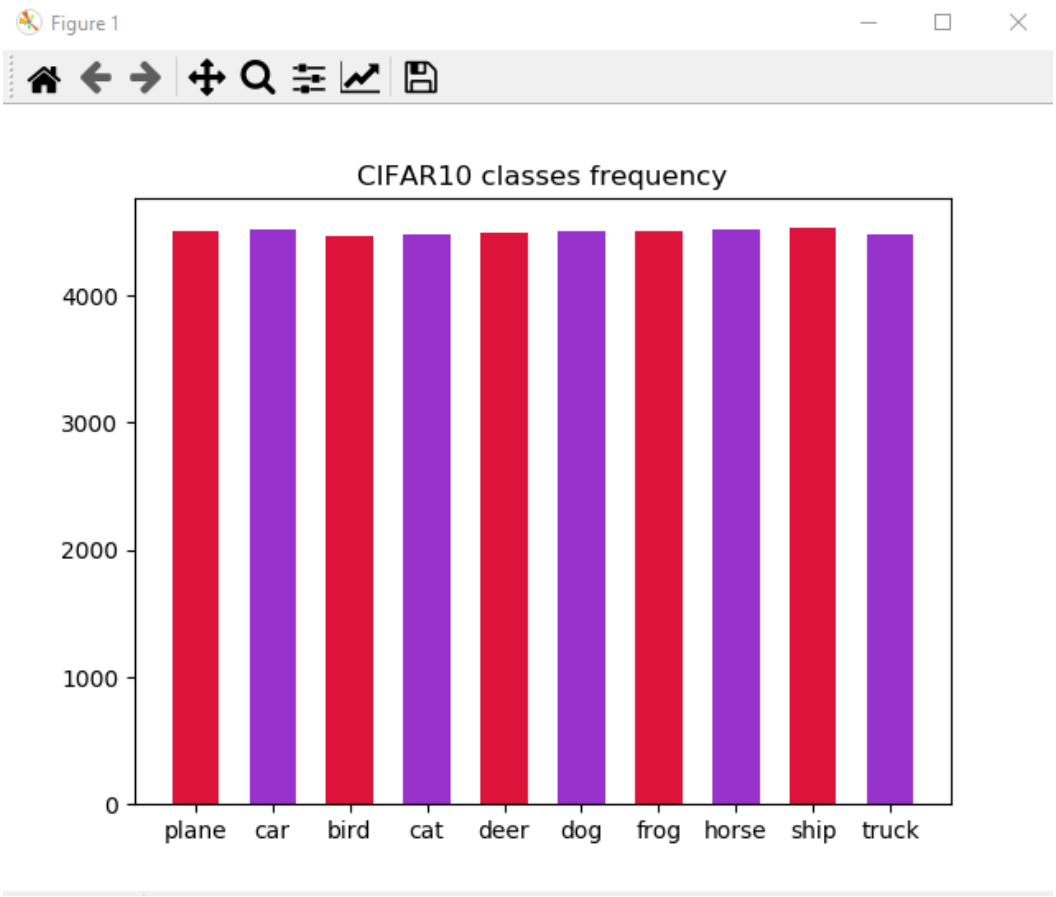


FIGURE 3.20 – distribution des images d'entrainement par classe

➤ train : lancement de l'apprentissage (affichage de la perte et de la précision)

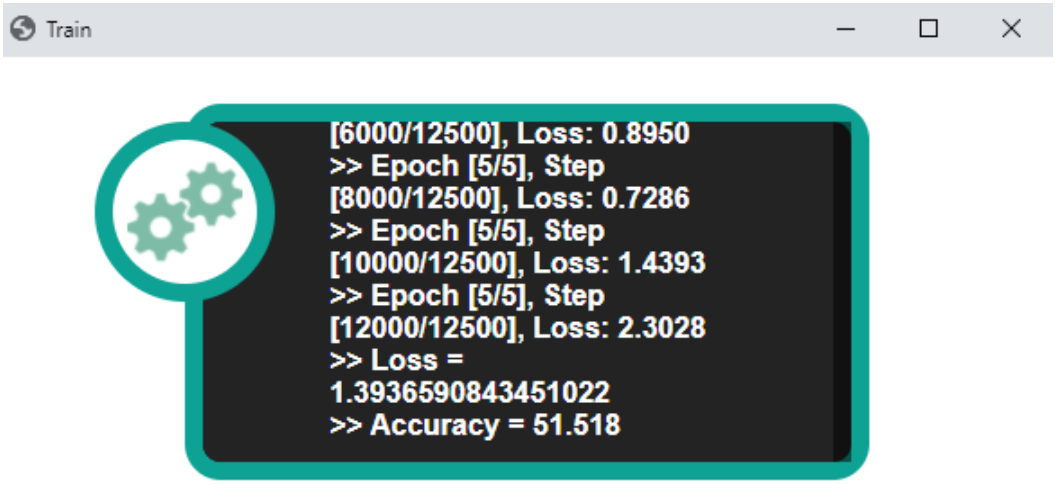


FIGURE 3.21 – La phase d'entrainement

➤ Graph : visualisation graphique de précision et de perte pendant la phase d'entrainement.

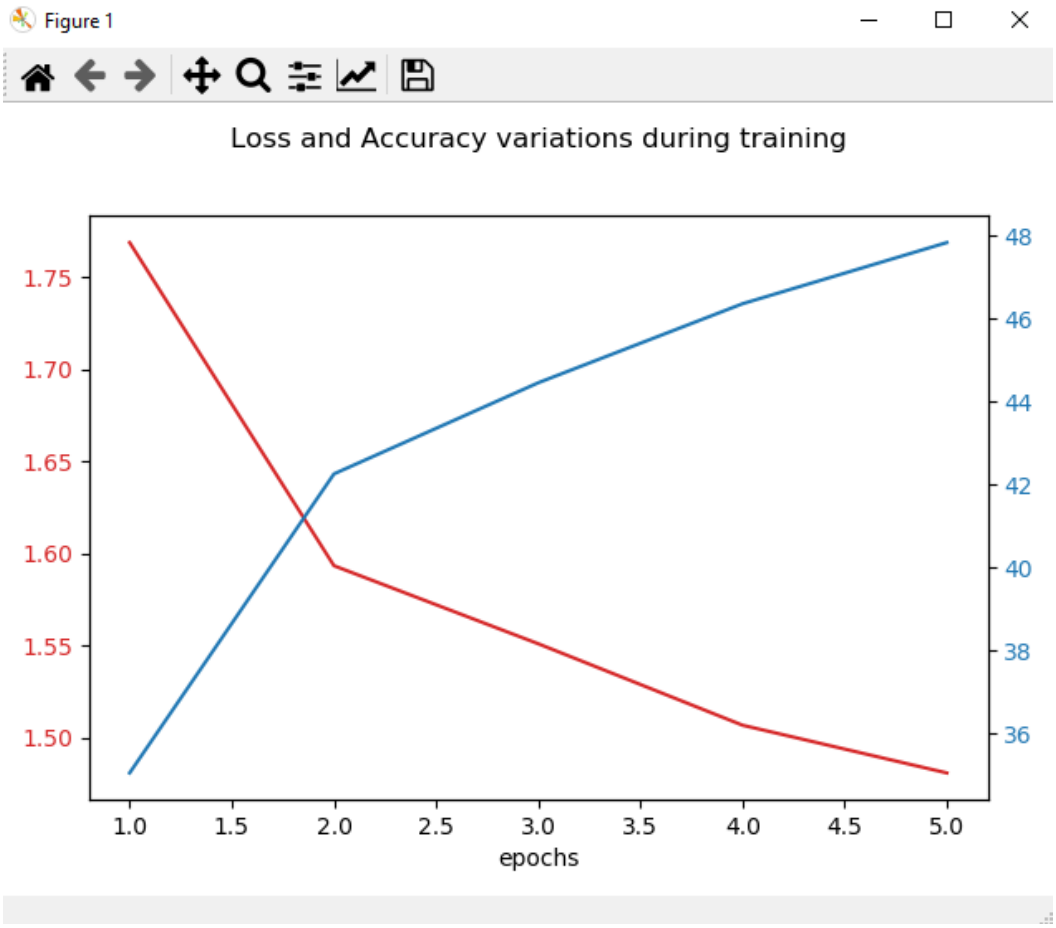


FIGURE 3.22 – La précision et la perte durant la phase d'entrainement

➤ Test : lancement du test MC Dropout (affichage de la précision et d'incertitude pour chaque classe).

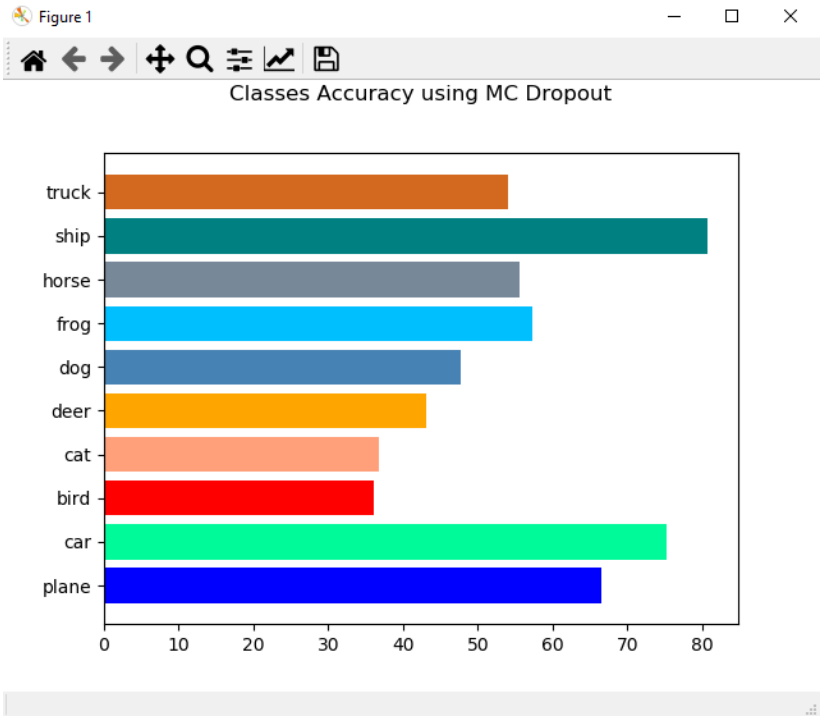


FIGURE 3.23 – La précision de chaque classe durant la phase du test : Application

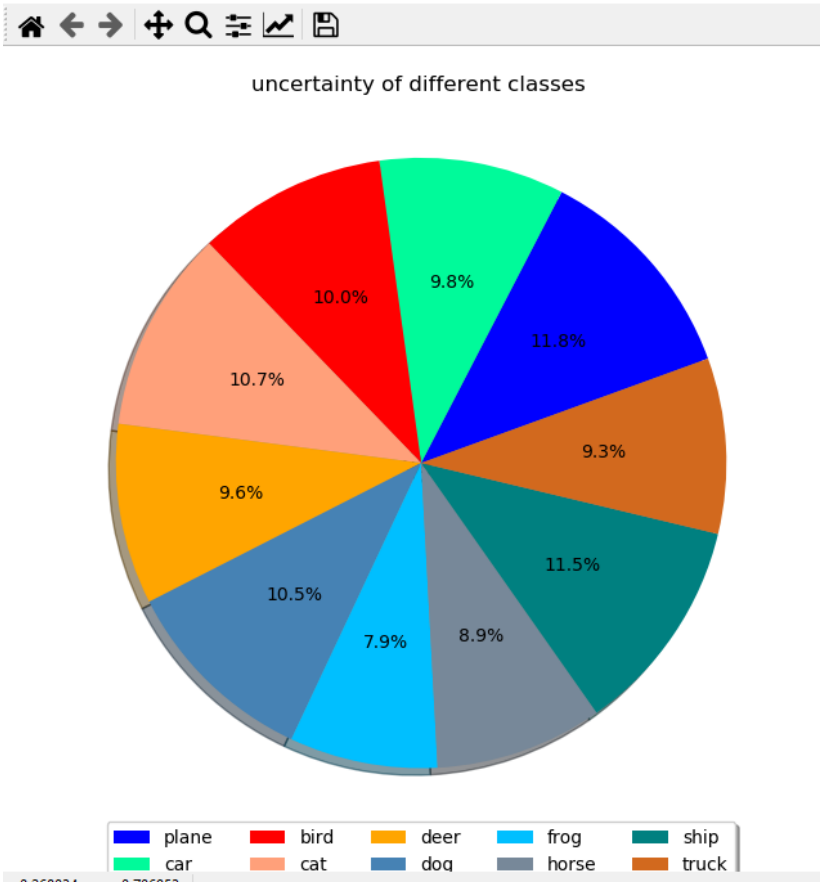


FIGURE 3.24 – Pourcentage d'incertitude de différentes classes : Application

Le code source de l'application et du rapport de mémoire rédigé en Latex sont disponibles dans le dépôt distant (Github) suivant :
https://github.com/merahsamia/PFE_BayesianDeepNetworksForBigDataMining

3.6 conclusion

Dans ce chapitre nous avons présenté les différentes expérimentations ainsi que notre méthodologie d'exécution et les différentes architectures des modèles utilisés, avant de présenter les résultats obtenus. Enfin, nous avons montré l'intégration de ces résultats dans notre application à travers une interface utilisateur.

Conclusion générale

L'explosion quantitative des données a obligé les chercheurs à trouver de nouvelles manières robustes de voir et d'analyser le monde des données, d'où la prise en compte des réseaux profonds bayésiens.

Le réseau profond bayésien offre une nouvelle approche basée sur la notion d'incertitudes qui répond à la question : "Quelle est la fiabilité de ma prédiction ?". Cependant, dans leur forme classique, les réseaux bayésiens étaient difficiles à implémenter de par la complexité spatiale et temporelle de la phase d'entraînement (respectivement espace de stockage et temps d'exécution nécessaire à un entraînement). Les désavantages techniques de ces méthodes ont longtemps mis à l'écart l'apprentissage profond bayésien de la révolution de l'intelligence artificielle dans laquelle nous vivons actuellement. Cependant, le MC Dropout remet à l'honneur ce pan de l'apprentissage automatique en s'adaptant directement à tout réseau déjà entraîné. Il semble aujourd'hui possible d'attribuer un degré de confiance à un réseau de neurones.

Dans le présent mémoire, nous avons d'abord défini et présenté ce que c'est l'apprentissage automatique et, de manière plus précise, les réseaux de neurones, avant d'exposer les différentes techniques d'apprentissage et notions liées au deep learning, ainsi que des généralités sur les réseaux bayésiens. Et ce, afin d'enlever l'ambiguïté quant à leurs différents concepts pour cerner de manière pertinente les possibilités d'exploitation de leurs techniques.

Nous avons ensuite effectué un travail de recherche ardu sur l'état de l'art des réseaux profonds bayésiens. Nous avons ainsi identifié les travaux les plus fiables ainsi que les différentes techniques et nouveaux concepts introduits et utilisés au fil des avancées dans la recherche, afin d'y trouver les plus performants et intéressants dans notre projet. Cependant, un des principaux freins au projet lors de cette phase a été le manque de ressources, l'intérêt

des chercheurs est assez récent, non pas seulement pour le big data d'ailleurs. Nous avons passé beaucoup de temps à lire et à étudier les publications et les articles pour voir ce qui se fait de mieux pour pouvoir concevoir notre propre modèle.

Nous avons été confrontés à quelques problèmes dans la phase d'implémentation. Parmi ces problèmes : l'absence de matériels assez conséquents pour nous permettre d'explorer et d'expérimenter davantage les nombreux nouveaux concepts et techniques introduits ces dernières années et façonnant le développement de la recherche. L'utilisation d'un CPU a fait que le temps d'exécution était trop couteux. Afin de régler ce problème on doit utiliser des réseaux de neurones plus profonds déployés sur un GPU au lieu d'un CPU sur des bases plus importantes.

Pendant ce PFE, et avec la fermeture des établissements au niveau national afin de prévenir la propagation du coronavirus, nous avons appris à travailler à distance avec l'établissement CERIST (Centre de recherche sur l'information scientifique et technique), ça nous a été bénéfique, car cela nous a permis de gagner en flexibilité du rythme et d'organisation et d'apprendre comment compter sur soi pour résoudre les problèmes au cas où ils se présentent et comment être méticuleux dans notre travail.

Nous pouvons dire à présent que notre objectif a été atteint, puisque l'application que nous avons réalisée satisfait les ambitions définies en premier lieu.

Toutefois, il est important de noter qu'il ne s'agit que d'un premier aperçu de ce que pourrait réellement être un réseau profond bayésien et que la solution proposée reste largement perfectible, on peut dire que ce travail n'est que dans sa version initiale mais ça nous a permis de mettre en pratique nos connaissances sur les réseaux de neurones et les réseaux bayésiens et d'en acquérir d'autres et le temps passé à lire des articles nous a servi d'une bonne initiation à la recherche.

Bibliographie

(**Alexandre Perrot, novembre 2017)** La visualisation d'information à l'ère du Big Data : résoudre les problèmes de scalabilité par l'abstraction multi échelle, thèse de doctorat L'UNIVERSITÉ DE BORDEAUX.

(**ALEXEI GRINBAUM, juin 2017)** Clefs - #64 - Les voix de la recherche VOYAGE AU COEUR DU BIG DATA, Big Data : de quoi parle-t-on ?

(**Corentin HARDY, Avril 2019)** Contribution au développement de l'apprentissage profond dans les systèmes distribués, thèse de doctorat, L'UNIVERSITE DE RENNES 1 COMUE UNIVERSITE BRETAGNE LOIRE.

(**Bernard ESPINASSE et Patrice BELLOT, 2017)** Introduction au Big Data Opportunités, stockage et analyse des mégadonnées, Réf. : H6040 V1.

(**Mat, 2012)** Mathieu, Gaetan, Benoit, Stéphane. Fouille des réseaux sociaux en ligne.Synthèse bibliographique sur le data mining sociale.2012.

(**Ork, 2012)** Orkhan Jafaro et Subhan Grismov. Machine Learning rapport de projet dans le cadre d'un master 2 informatique-université de Franche-Comté, France 2012.

(**Rémi Connesson, 2018)** Le Deep Learning, comment fonctionne l'algorithme qui a bousculé le monde de l'intelligence artificielle. Le blog de la School of AI en France.

(**McCulloch, W.S. & Pitts, 1943)** McCulloch, W.S. & Pitts, W. (1943). Bulletin of Mathematical Biophysics.

(**Rosenblatt, 1958)** Rosenblatt, F. (1958) The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain. Cornell Aeronautical Laboratory, Psychological Review, 65, 386-408.

(**Chloé-Agathe Azencott, 2019)** Utilisez des modèles supervisés non linéaires. OpenClassrooms.

(**STEPHENE TUFFERY, 2019)** Big Data, Machine Learning et apprentissage profond.

(**Olivier Ezratty, Octobre 2017)** Les usages de l'intelligence artificielle.

(**Ian Goodfellow et al, 2016)** Ian Goodfellow, Yoshua Bengio and Aaron Courville .Deep Learning. Convolutional Neural Networks, page 329.

(**Andrej Karpathy, 2019)** Stanford CS class CS231n : Convolutional Neural Networks for Visual Recognition.

(**Radford et al, 2015)** Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv :1511.06434.

(**Bengio et al., 2014)** Bengio, Y., Laufer, E., Alain, G., & Yosinski, J. (2014, January). Deep generative stochastic networks trainable by backprop. In International Conference on Machine Learning (pp. 226-234).

(**Giorgio Roffo,2017)** Ranking to Learn and Learning to Rank : On the Role of Ranking in Pattern Recognition Applications. Thesis for : European Doctor of Philosophy.

(**Hasna Njah et al, 2016)** Deep Bayesian network architecture for Big Data mining

(**David Bellot, 2002)** Fusion de données avec des réseaux bayésiens pour la modélisation des systèmes dynamiques et son application en télémédecine Doctorat de l'université Henri Poincaré - Nancy 1

(**Olivier Francois, 2006)** De l'identification de structure de réseaux bayésiens à la reconnaissance de formes à partir d'informations complètes ou incomplètes.

(**Pearl, 1985)** J. Pearl et A. Paz. GRAPHOIDS : A graph-based logic for reasoning about relevance relations. Rapport technique (Cognitive Systems Laboratory, University of California, Los Angeles, 1985.

(**Chong Wang et al, 2011)** Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientic articles. In KDD, pages 448 ?456, 2011.

(**chr, 12)** Christophe Thovex, Réseaux de compétences de l'analyse des réseaux sociaux à l'analyse prédictive de connaissances. Artificial intelligence, université de Nantes, France 2012

(**Kostadin Georgiev et al, 2013)** Kostadin Georgiev and Preslav Nakov. A non-iid framework for collaborative filtering with restricted Boltzmann machines. In ICML, pages 1148 ?1156, 2013.

(**Naïm et al ,2007)** Naïm, P., Wuillemin, P., Leray, P., Pourret, O., Becker, A., Réseaux bayésiens, 3e éd., Eyrolles, Paris, 2007, 424 p.

(**Naïm et al ,2008)** Réseaux bayésiens, Paris, Eyrolles

(**Yiping Du, 2019)** Bayesian Deep Convolutional Neural Network for SMS Quality Analysis.

(**Pas,2009)** Pascal Germain. Algorithmes d'apprentissage automatique inspirés de la théorie PAC-Bayes, Faculté des sciences et de Génie, Université Laval, Québec, Canada .2009

(**Ruslan Salakhutdinove et al, 2007)** Ruslan Salakhutdinove, Andriy Mnih, and Geoffrey E. Hinton. Restricted Boltzmann machines for collaborative ?ltering. In ICML, pages 791-798, 2007.

(**Sachs et al, 2005)** « Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data », Science, vol. 308, nº 5721, p. 523-529.

(**Tristan Stérin ,2016)** Tristan Stérin : Réseaux de neurones récurrents et mémoire : application à la musique.

(**Kendall et Gal ,2017)** Alex Kendall, Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision ?

(**Raanan Y. Rohekar et al, 2018)** Raanan Y. Rohekar, Shami Nisimov, Yaniv Gurwicz, Guy Koren Gal Novik, 2018 : Constructing Deep Neural Networks by Bayesian Network Structure Learning.

(**Umara Zafar et al, 2019)** Umara Zafar, Mubeen Ghafoor, Tehseen Zia, Ghufran Ahmed, Ahsan Latif, Kaleem Razzaq Malik and Abdullahi Mohamud Sharif. Face recognition with Bayesian convolutional networks for robust surveillance systems. EURASIP Journal on Image and Video Processing, EURASIP Journal on Image and Video Processing.

(**Gal et Ghahramani, 2015)** Yarin Gal, Zoubin Ghahramani. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference.

(**Alex Kendall et al., 2017)** Alex Kendall, Vijay Badrinarayanan, Roberto Cipolla. Bayesian SegNet : Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding.

(**Gal et Ghahramani, 2016)** Yarin Gal, Zoubin Ghahramani. Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning.

(**XIAO SUN et al, 2019)** Xiao Sun, Jia Li, Xing Wei, Changliang Li, and Jianhua Tao. 2019. Emotional Conversation Generation Based on a Bayesian Deep Neural Network. ACM Trans. Inf. Syst. 38, 1, Article 8 (December 2019)

(**Gal et al, 2017)** Yarin Gal, Riashat Islam, Zoubin Ghahramani. Deep Bayesian Active Learning with Image Data.

(**Shen et al, 2018)** Yanyao Shen, Hyokun Yun, Zachary Chase Lipton, Yakov Kronrod. Deep Active Learning for Named Entity Recognition.

(**Blundell et al, 2015)** Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, Daan Wierstra. Weight Uncertainty in Neural Networks.

(**Deshpande, 2016)** DESHPANDE, Adit, 2016. A Beginner ?s Guide To Understanding Convolutional Neural Networks Part 2. In : [en ligne]. 17 2016. [Consulté le 21 aout 2020]. Disponible à l ?adresse : https://adeshpande3.github.io/A-Beginner%27s-Guide-To-UnderstandingConvolutional-Neural-Networks-Part-2/.

(**Srivastava et al, 2014)** Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout. A Simple Way to Prevent Neural Networks from Overfitting.

(**Der Kiureghian et Ditlevsen, 2009)** Aleatory or epistemic ? does it matter ? Structural Safety, 31(2) :105 ?112.

(**Ronald Seoh, 2020)** Qualitative Analysis of Monte Carlo Dropout, University of Massachusetts Amherst.

(**Gal, 2016)** Yarin Gal. Uncertainty in deep learning. PhD thesis, PhD thesis, University of Cambridge, 2016.

Webographie

(1) LE DEEP LEARNING PAS À PAS : LES CONCEPTS (1/2). https://www.technologies-ebusiness.com/enjeux-et-tendances/le-deep-learning-pas-a-pas. [Consulté : 06/08/2020].

(2) Harry Pommier, 2020. [Fiche mémo] Réseau de neurones profond. https://medium.zenika.com/fiche-m%C3%A9mo-r%C3%A9seau-de-neurones-profond-66f6bf48a472. [Consulté : 25/08/2020].

(3) Modèles basés sur le réflex : apprentissage automatique https://stanford.edu/~shervine/l/fr/teaching/cs-221/pense-bete-modeles-reflex. [Consulté : 08/08/2020].

(4) La rétropropagation du gradient. https://dataanalyticspost.com/Lexique/retropropagation/#:~:text=La%20r%C3%A9tropropagation%20du%20gradient%20de,dans%20une%20base%20d'apprentissage. [Consulté : 12/08/2020].

(5) Thierry Maubant. Qu'est-ce que le surapprentissage ? https://www.actuia.com/faq/quest-ce-que-le-surapprentissage/. [Consulté : 07/08/2020].

(6) Artificial intelligence and the rise of economec inequality https://www.datascience.us/artificial-intelligence-rise-economic-inequality/. [Consulté : 15/08/2020].

(7) Afshine Amidi et Shervine Amidi. (2018). Pense-bête de réseaux de neurones convolutionnels. https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels.

(8) Convolutional Neural Networks (CNN) : Step 3 - Flattening. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening. [Consulté : 22/08/2020].

(9) Par Jean-Pierre Mistral.(2018) BIG DATA, INTELLIGENCE ARTIFICIELLE, APPRENTISSAGE AUTOMATIQUE ET

LE RGPD. https://www.village-justice.com/articles/big-data-intelligence-artificielle-apprentissage-automatique-rgpd,26838.html. [Consulté : 19/08/2020].

(10) Yarin Gal. What My Deep Model Doesn't Know http://mlg.eng.cam.ac.uk/yarin/blog_3d801aa532c1ce.html. [Consulté : 28/08/2020].

(11) Robin Camarasa. Bayesian Deep Learning : Soyez sûr de vos incertitudes. https://www.datagenius.fr/post/bayesian-deep-learning-soyez-sur-de-vos-incertitudes. [Consulté : 26/08/2020].

(12) Gábor Vecsei. Experiments with Monte-Carlo dropout for uncertainty estimation. https://www.gaborvecsei.com/Monte-Carlo-Dropout/. [Consulté : 26/08/2020].

(13) Deep learning Dropout technical analysis. https://www.programmersought.com/article/31754654312/. [Consulté : 28/08/2020].

(14) Python. https://fr.wikibooks.org/wiki/Programmation_Python/Introduction. [Consulté : 06/08/2020].

(15) Anaconda. https://www.anaconda.com/what-is-anaconda/. [Consulté : 06/08/2020].

(16) Pytorch. https://datafranca.org/wiki/PyTorch. [Consulté : 06/08/2020].

(17) Numpy. https://openclassrooms.com/fr/courses/4452741-decouvrez-les-librairies-python-pour-la-data-science/4740941-plongez-en-detail-dans-la-librairie-numpy. [Consulté : 06/08/2020].

(18) Matplotlib https://matplotlib.org/. [Consulté : 06/08/2020].

(19) HTML Tutorial. https://www.w3schools.com/html/. [Consulté : 04/09/2020].

(20) CSS Tutorial. https://www.w3schools.com/css/. [Consulté : 04/09/2020].

(21) JavaScript Tutorial https://www.w3schools.com/js/. [Consulté : 04/09/2020].

(22) Create HTML User Interface for Python using Eel Library https://medium.com/wronmbertech/create-html-user-interface-for-python-using-eel-library-bab101cc0f99. [Consulté : 04/09/2020].

(23) Pycharm https://www.jetbrains.com/fr-fr/pycharm/. [Consulté : 06/08/2020].

(24) Jupyter. https://jupyter-notebook.readthedocs.io/en/stable/. [Consulté : 06/08/2020].

(25) Cifar 10 http://www.cs.toronto.edu/~kriz/cifar.html. [Consulté : 06/08/2020].

(26) Useful Plots to Diagnose your Neural Network. https://towardsdatascience.com/useful-plots-to-diagnose-your-neural-network-521907fa2f45. [Consulté : 28/08/2020].