République Algérienne Démocratique Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université d'Ibn Khaldoun – Tiaret

Faculté des Mathématiques et de l'Informatique

Département d'Informatique

Thème

Recherche et classification des images en utilisant les modèles pré-trainés des CNNs

Pour l'obtention du diplôme de Master II

Spécialité : **Génie Logiciel**

Réalisé par : - **BOUABDELLI Keltoum**

- **LARBI Hanane**

Soutenu le 29/12/2020 à Tiaret devant le jury composé de :

Mr TALBI Omar	MCB	Université de Tiaret	Président
Mr MEGHAZI Hadj Madani	MAA	Université de Tiaret	Examinateur
Mr MERATI Medjeded	MCB	Université de Tiaret	Encadreur

Année universitaire 2019-2020

Dédicace

Nous dédions notre travail à :

A tous les membres de la famille et à tous les amis.

Remerciements

- *Nous tenons à remercier tout d'abord notre encadreur : Mr **MERATI Medjeded**.*

- *Nous remercions également les membres de jury Mr **TALBI Omar** et Mr **MEGHAZI Hadj Madani** d`avoir accepté de juger notre travail.*

- *Nous remercions toutes les personnes qui nous ont soutenues de près ou de loin dans la réalisation de ce travail*

Résumé

La recherche d'images similaires à une image requête est une problématique connue dans le domaine de traitement d'images.

Dans ce contexte, nous avons proposé un système de recherche d'images combinant les mesures de similarité avec la classification des images basée sur l'apprentissage automatique.

A cet effet, nous avons choisi d'utiliser un modèle de réseau de neurones convolutif (CNN) « Convolutional Neural Networks » dans la classification des images et la distance euclidienne comme mesure de similarité entre l'image requête et les images de la base classées par le CNN comme étant de la même classe que l'image requête. Afin de valider notre système proposé, nous avons appliqué ce dernier sur la base d'images CIFAR-10.

Mots clés : Recherche, Classification, Images similaires, CNN, Distance euclidienne, CIFAR-10.

Sommaire

Liste des figures

Liste des tables

Liste des abréviations

ANN : Artificial Neural Network

API : Application Programming Interface

CNN : Convolutional Neural Networks

ConvNet : Convolutional Network

DenseNet : Dense Convolutional Network

FC : fully connected

ILSVRC : ImageNet Large Scale Visual Recognition Challenge

ImageNet : Image Network

MLP : Multi Layers Perceptron

MNIST : Modified National Institute of Standards and Technology

ResNet : Residual Network

PCA : Principal Component Analysis

VGG : Visual Geometry Group

Introduction Générale

Le nombre de bases d'images numériques est actuellement en augmentation rapide, et la taille du contenu multimédia créé et stocké chaque jour croit significativement. En même temps un besoin d'accès rapide à des informations faiblement structurées (images, musiques, etc.) se précise jour après jour.

Dans ce contexte, la recherche d'images similaires à une image requête de l'utilisateur constitue une problématique très connue dans le domaine de traitement des images.

A cet effet, le besoin à des systèmes de recherche d'images répondant aux demandes des utilisateurs se fait sentir de plus en plus.

Par conséquent, nous avons, à travers ce travail, proposé et réalisé un outil capable de rechercher des images similaires à une image requête.

Notre approche est basée sur une combinaison d'une classification d'images utilisant les modèles CNNs pré-trainés et d'une mesure de similarité à savoir la distance euclidienne.

Après avoir classé l'image requête et les images de la base CIFAR [14] en utilisant le modèle CNN, nous sélectionnons uniquement celles ayant la même classe que l'image requête. Ensuite, afin de perfectionner le résultat de la classification, nous utilisons la distance euclidienne comme mesure de similarité entre l'image requête et les images de CIFAR retournées par la classification. La moyenne, l'écart-type et la valeur de score (cette dernière est attribuée par le CNN à chaque image) sont calculés pour caractériser les images et calculer la distance euclidienne. Finalement, la liste des images est triée de manière descendante selon la distance. Ladite liste ou ses premières images sont retournées comme résultat de la recherche des images similaires.

Après l'introduction générale, notre mémoire est organisé comme suit :

Le chapitre un, donne les définitions des différents concepts à savoir : la classification des images et ses différents types, les réseaux de neurones et l'apprentissage automatique.

Le chapitre deux est consacré à la description de l'apprentissage profond et ses différentes techniques.

Dans le troisième chapitre, nous détaillons la partie expérimentale de notre travail dans laquelle nous décrirons nos modèles proposés pour la recherche et la classification des images et les techniques utilisées pour la recherche d'image.

Enfin, nous concluons ce mémoire et proposons des perspectives à notre travail dans la conclusion générale.

Chapitre 1
Classification des images

Introduction

Avec le développement rapide de l'Internet, de plus en plus d'informations sur les images sont stockées sur Internet. L'image est devenue aussi un important support d'informations réseau que le texte. Dans ce contexte, il est très important d'utiliser des systèmes pour classer et reconnaître intelligemment ces images et les faire mieux servir aux utilisateurs.

La reconnaissance d'image, en particulier (reconnaissance de catégorie d'objet) dans les scènes naturelles, est une compétence unique des êtres humains. Dans un environnement naturel complexe et en un coup d'œil, l'humain peut facilement identifier des objets ou une catégorie spécifique d'objets (articles ménagers, oiseaux, etc.). Cependant, il y a encore beaucoup de questions qui restent posées sur la façon dont les êtres humains font cela et comment appliquer ces technologies connexes aux ordinateurs afin qu'ils aient une intelligence humanoïde.

I. Définition

La classification des images consiste à attribuer automatiquement une classe à une image. On retrouve ainsi la classification d'objets, de scènes, de textures, la reconnaissance de visages, d'empreintes digitale et de caractères. Il existe deux principaux types d'apprentissage: l'apprentissage supervisé et l'apprentissage non-supervisé. Dans notre travail, nous nous intéressons à l'approche supervisée.

Un modèle de classification d'images reçoit un ensemble d'images dans une classe spécifique. Sur la base de cet ensemble, l'algorithme apprend à quelle classe appartiennent les images d'apprentissage, puis peut prédire la classe correcte des futures entrées d'image et peut même mesurer la précision des prédictions.

II. Les motivations de la classification des images

La classification des images consiste à répartir systématiquement des images selon des classes établies au préalable. Classer une image, c'est lui faire correspondre une classe, marquant ainsi sa parenté avec d'autres images.

En général, reconnaître une image est une tâche aisée pour un humain qui au fil de son existence, a acquis des connaissances qui lui permettent de s'adapter aux variations résultant de différentes conditions d'acquisition. Il lui est par exemple relativement simple de

reconnaître un objet dans plusieurs orientations et/ou partiellement caché par un autre de près ou de loin et selon diverses illuminations.

Toutefois, les progrès technologiques en termes d'acquisition d'images (microscopes, caméras, capteurs) et de stockage engendrent des bases de données riches en information et multiplient les domaines d'applications, il devient alors difficile pour l'humain d'analyser le nombre important d'images où le temps requis, le caractère répétitif de la tâche et la concentration nécessaire deviennent problématiques. Toutefois, ces problématiques ne sont pas forcément une tâche aisée pour un programme informatique pour lequel une image n'est qu'un ensemble de valeurs numériques.

III. L'objectif de la classification

L'objectif de la classification d'images est d'élaborer un système capable d'affecter automatiquement une image à une classe. Ainsi, ce système permet d'effectuer une tâche d'expertise qui peut s'avérer coûteuse à acquérir pour un être humain en raison notamment de contraintes physiques comme la concentration, la fatigue ou le temps nécessaire à un volume important d'images.

IV. Méthodes de classification

De nombreuses méthodes classiques ont été consacrés, elles peuvent être séparées en deux grandes catégories : les méthodes de classification supervisée et les méthodes de classification non supervisée.

1. Méthodes supervisées

L'objectif de la classification supervisée est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets. On dispose au départ d'un échantillon dit d'apprentissage dont les classes de ses éléments sont connues. Cet échantillon est utilisé pour l'apprentissage des règles de classification.

Il est nécessaire d'étudier la fiabilité de ces règles en évaluant les cas de sous apprentissage ou de sur-apprentissage (complexité du modèle). On utilise souvent un deuxième échantillon indépendant, dit de validation ou de test [1].

2. Méthodes non supervisées

Ces méthodes ne nécessitent aucun apprentissage et aucune tâche préalable d'étiquetage manuel. Elle consiste à représenter un nuage de points d'un espace quelconque

en un ensemble de groupes appelés clusters. Un « **Cluster** » est une collection d'objets qui sont « **similaires** » entre eux et qui sont « dissemblables » aux objets appartenant à d'autres groupes [2]. Cette manière de procéder est généralement liée au domaine de l'analyse des données dont la méthode Analyse en Composantes Principales (ACP) est un bon exemple.

V. Apprentissage automatique

Les méthodes manuelles se sont avérées très difficiles à appliquer pour des tâches en apparence très simples comme la classification des images, la reconnaissance d'objets dans les images ou la reconnaissance vocale. Les données venant du monde réel, les échantillons d'un son ou les pixels d'une image sont complexes, variables et entachées de bruit.

Pour une machine, une image est un tableau de nombres indiquant la luminosité (ou la couleur) de chaque pixel, et un signal sonore est une suite de nombres indiquant la pression de l'air à chaque instant.

Comment une machine peut-elle transcrire la suite de nombres d'un signal sonore en série de mots tout en ignorant le bruit ambiant, l'accent du locuteur et les particularités de sa voix ? Comment une machine peut-elle identifier un chien ou une chaise dans le tableau de nombres d'une image quand l'apparence d'un chien ou d'une chaise et des objets qui les entourent peut varier infiniment ?

Il est virtuellement impossible d'écrire un programme qui fonctionnera de manière robuste dans toutes les situations. C'est là qu'intervient l'apprentissage automatique (que l'on appelle aussi apprentissage machine). C'est l'apprentissage automatique qui anime les systèmes de toutes les grandes entreprises d'Internet. Elles l'utilisent depuis longtemps pour filtrer les contenus indésirables, ordonner des réponses à une recherche, faire des recommandations, ou sélectionner les informations intéressantes pour chaque utilisateur [3].

Le problème de l'approche classique de la classification des images est qu'un bon extracteur de caractéristiques est très difficile à construire, et qu'il doit être repensé pour chaque nouvelle application.

C'est là qu'intervient l'apprentissage profond ou Deep Learning en anglais. C'est une classe de méthodes dont les principes sont connus depuis la fin des années 1980, mais dont l'utilisation ne s'est vraiment généralisée que depuis 2012, environ.

L'idée est très simple : le système entraînable est constitué d'une série de modules, chacun représentant une étape de traitement. Chaque module est entraînable, comportant des paramètres ajustables similaires aux poids des classifieurs linéaires. Le système est entraîné de bout en bout : à chaque exemple, tous les paramètres de tous les modules sont ajustés de

manière à rapprocher la sortie produite par le système de la sortie désirée. Le qualificatif profond vient de l'arrangement de ces modules en couches successives.

Pour pouvoir entraîner le système de cette manière, il faut savoir dans quelle direction et de combien ajuster chaque paramètre de chaque module. Pour cela, il faut calculer un gradient, c'est-à-dire pour chaque paramètre ajustable, la quantité par laquelle l'erreur en sortie augmentera ou diminuera lorsqu'on modifiera le paramètre d'une quantité donnée. Le calcul de ce gradient se fait par la méthode de rétropropagation, pratiquée depuis le milieu des années 1980.

Dans sa réalisation la plus commune, une architecture profonde peut être vue comme un réseau multicouche d'éléments simples, similaires aux classifieurs linéaires, interconnectés par des poids entraînables. C'est ce qu'on appelle un réseau neuronal multicouches.

Ce qui fait l'avantage des architectures profondes, c'est leur capacité d'apprendre à représenter le monde de manière hiérarchique. Comme toutes les couches sont entraînables, nul besoin de construire un extracteur de caractéristiques à la main puisque l'entraînement s'en chargera [4].

VI. Conclusion

Nous avons consacré ce chapitre à la présentation des notions de la classification ainsi que leurs intérêts dans le domaine d'imagerie et parlé aussi de l'utilisation de l'apprentissage automatique dans ce domaine. Dans le deuxième chapitre, nous détaillerons la notion d'apprentissage profond (Deep Learning) et les réseaux de neurones et plus précisément l'utilisation des réseaux de neurones convolutionnels dans la classification des images.

Chapitre 2
Apprentissage profond

Introduction

Apprentissage profond, Deep Learning en anglais, ou encore « rétropropagation de gradient » … ces termes, quasi synonymes, désignent des techniques d'apprentissage machine (machine learning), une sous-branche de l'intelligence artificielle qui vise à construire automatiquement des connaissances à partir de grandes quantités d'information.

Le Deep Learning est capable d'apprendre par lui-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées.

Les succès qu'enregistrent ces techniques leurs confèrent un rôle essentiel dans le monde contemporain, où elles apparaissent être à l'origine d'innombrables applications pratiques (reconnaissance des visages et de la parole, voiture autonome, etc.).

Dans ce qui suit, nous présenterons les notions en relation avec l'apprentissage profond.

I. Définition

Le Deep Learning est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires, il permet aux modèles de calcul composés de plusieurs couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction. Ces méthodes sont à l'origine de beaucoup de travaux dans plusieurs domaines d'application tels que la reconnaissance vocale, la reconnaissance visuelle d'objets, la détection d'objets et de nombreux autres domaines tels que la découverte de médicaments et la génomique. Le Deep Learning explore de grands ensembles de données en utilisant l'algorithme de rétro propagation pour indiquer comment une machine doit modifier ses paramètres internes utilisés pour calculer la représentation dans chaque couche à partir de la représentation dans la couche précédente.

Les réseaux de convolution profonds ont permis des percées dans le traitement des images, de la vidéo, de la parole et de l'audio, tandis que les réseaux récurrents ont mis en lumière des données séquentielles telles que le texte et la parole [5].

II. Domaines d'application

Le Deep Learning change notre façon de voir les technologies et suscite actuellement beaucoup d'intérêt.

Il est prévu que de nombreuses applications d'apprentissage profond affecteront notre vie dans un proche avenir. En fait, ils ont déjà un impact.

Au cours des cinq à 10 prochaines années, les outils de développement d'apprentissage profond, les bibliothèques et les langages deviendront des composants standards de chaque boîte à outils de développement logiciel. L'apprentissage profond est très exploité dans des domaines d'application tels que : les voitures autonomes, les soins de santé (diagnostic du cancer du sein ou de la peau ; …), recherche vocale, ajout automatique de sons à des films silencieux, traduction automatique, génération automatique de texte, génération d'écriture automatique, classification des images, génération automatique de légendes d'images, colorisation automatique, la publicité, prédiction des tremblements de terre [6].

III. Réseau de Neurones Artificiel

Un Réseau de Neurones Artificiels ou Artificial Neural Network en anglais (ANN) est inspiré du fonctionnement du cerveau humain.

1. Fonctionnement des ANNs

La conception des ANNs s'appuie sur la structure des neurones biologiques du cerveau humain.

Les ANNs peuvent être décrits comme des systèmes composés d'au moins deux couches de neurones - une couche d'entrée et une couche de sortie - et comprenant généralement des couches intermédiaires (« *hiddenlayers* »). Plus le problème à résoudre est complexe, plus le ANN doit avoir de couches. Chaque couche contient un grand nombre de neurones spécialisés.

2. Les différents types d'ANNs

On fait appel à différentes structures de réseaux de neurones artificiels en fonction de la méthode d'apprentissage utilisée et de l'objectif recherché. Dans cette section, nous en introduirons le modèle perceptron et ainsi que le modèle à propagation avant.

1.1 Perceptron

À l'origine, la forme la plus simple de réseau neuronal artificiel était composée d'un seul neurone modifié par des pondérations et doté d'une valeur seuil. Désormais, le terme « Perceptron » désigne aussi les réseaux à propagation avant et à couche unique.

1.2 Réseaux de neurones à propagation avant

Un ANN à propagation avant (figure 1) ne peut transmettre l'information que dans un unique sens de traitement**.** Les réseaux peuvent être monocouches, c'est-à-dire constitués

uniquement de couches d'entrée et de sortie, ou multicouches, c'est-à-dire disposant d'un certain nombre de couches cachées.

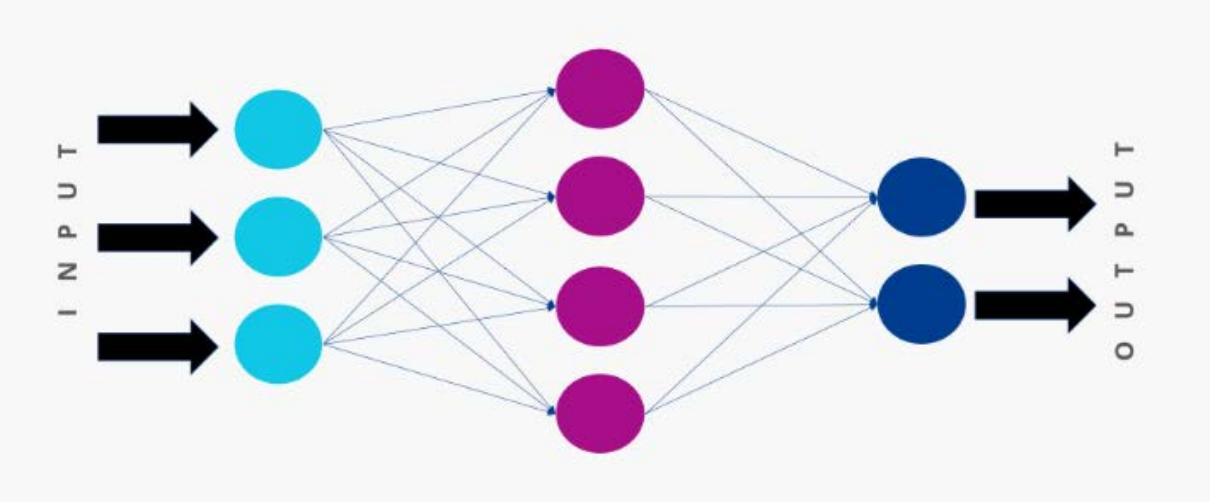


Figure 1: Réseaux de neurones à propagation avant.

VII. Les réseaux de neurones convolutifs

Un réseau neuronal convolutif ou Convolutional Neural Network (CNN) en anglais est un type de réseau multicouche. Il est composé d'un minimum de cinq couches. Il est procédé sur chacune de ces couches à une reconnaissance de motif. Le résultat obtenu sur chaque couche est transmis à la couche suivante. Ce type de réseau neuronal artificiel est utilisé en matière de reconnaissance d'images [7].

Il existe un grand nombre d'architectures profondes. La plupart d'entre elles sont dérivées de certaines architectures originales. Nous avons basé notre travail sur les CNNs du fait que ces derniers sont bien adaptés à la classification des images.

Les CNNs sont directement inspirés du cortex visuel des vertébrés. Un CNN, appelé aussi ConvNet « Convolutional Network ».

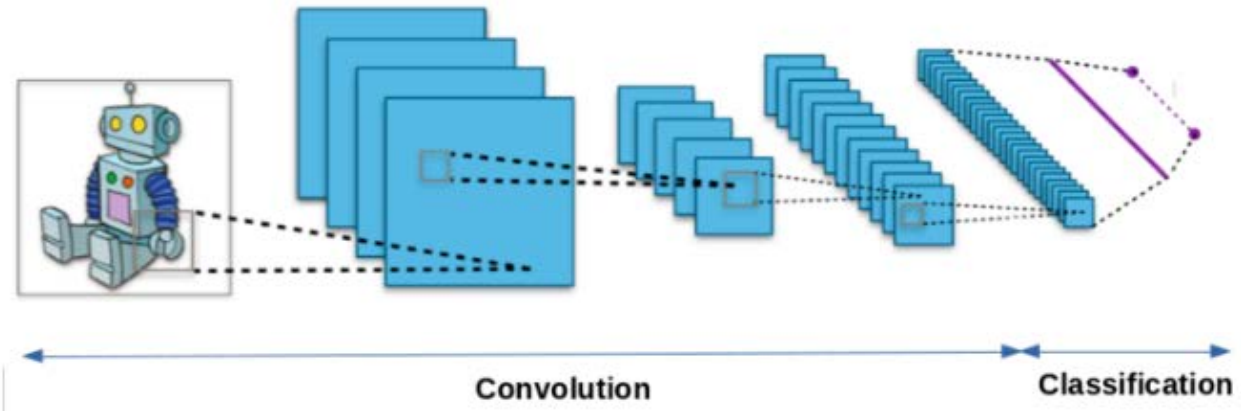


Figure 2: Architecture d'un CNN.

Comme illustré dans la figure 2, on distingue deux parties dans une architecture d'un CNN, une première partie que l'on appelle la partie convolutive du modèle et la seconde partie, appelée partie de classification qui correspond à un modèle MLP (Multi Layers Perceptron).

3. Les couches de CNNs

Une architecture CNN est formée d'un empilement de couches de traitement indépendantes : Couche de convolution, Couche pooling, couche ReLu et couche Loss.

3.1 La couche de convolution (CONV)

La convolution est un outil mathématique simple largement utilisé dans le traitement d'image, et plus particulièrement dans la reconnaissance d'objets**.**

La convolution agit comme un filtrage. On définit une taille de fenêtre qui va se balader à travers toute l'image. Au tout début de la convolution, la fenêtre est positionnée tout en haut à gauche de l'image puis elle se décale d'un certain nombre de cases vers la droite et lorsqu'elle arrive au bout de l'image, elle se décale d'un pas vers le bas ainsi de-suite jusqu'à ce que le filtre ait parcourue la totalité de l'image en se servant des valeurs présentes dans le filtre à chaque pas (figure 3).

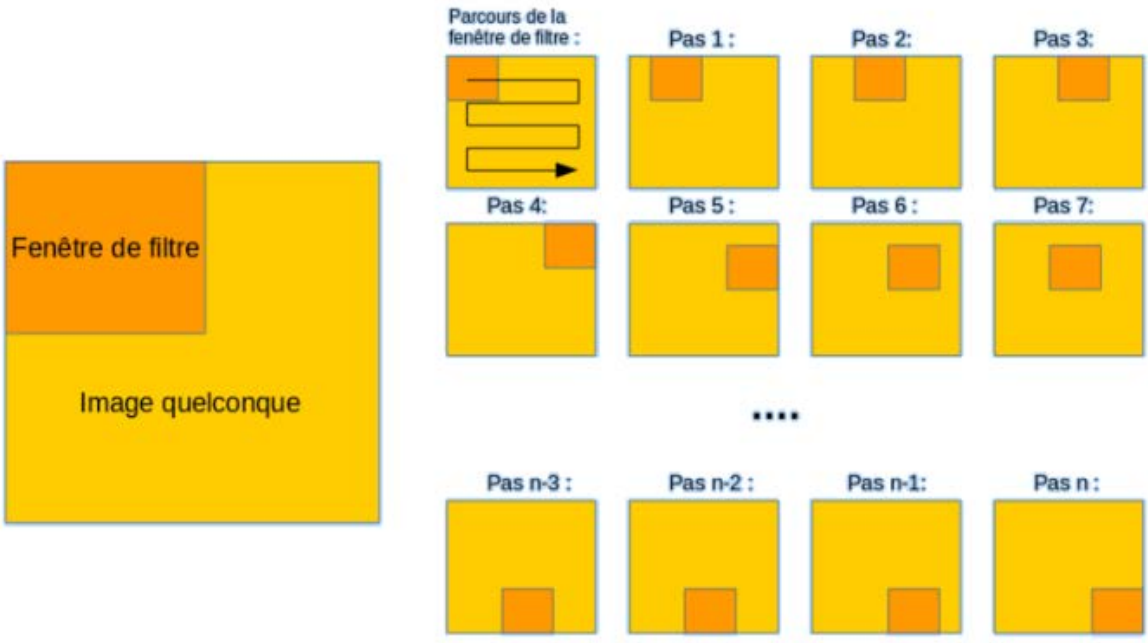


Figure 3: Schéma du parcours de la fenêtre de filtre sur l'image [8].

3.2 Couche de Pooling

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs cartes de caractéristiques (features maps), et applique à chacune d'entre elles l'opération de Pooling.

L'opération de Pooling (ou sub-sampling) consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Pour cela, on découpe l'image en cellules régulière, puis on garde au sein de chaque cellule la valeur maximale dans le cas d'un MaxPooling. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille 2 × 2 pixels qui ne se chevauchent pas, ou des cellules de taille 3 × 3 pixels, distantes les unes des autres d'un pas de 2 pixels (qui se chevauchent donc). On obtient en sortie le même nombre de feature maps qu'en entrée, mais celles-ci sont bien plus petites (figure 4).

Il est courant d'insérer périodiquement une couche Pooling entre les couches Conv successives dans une architecture ConvNet. Sa fonction est de réduire progressivement la taille spatiale de la représentation afin de réduire la quantité de paramètres et de calculs dans le réseau, et donc de contrôler également le sur-ajustement. La couche de Pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne spatialement, en utilisant l'opération MAX dans le cas de MaxPooling.

Ainsi, la couche de Pooling rend le réseau moins sensible à la position des features : le fait qu'une feature se situe un peu plus en haut ou en bas, ou même qu'elle ait une orientation légèrement différente ne devrait pas provoquer un changement radical dans la classification de l'image.

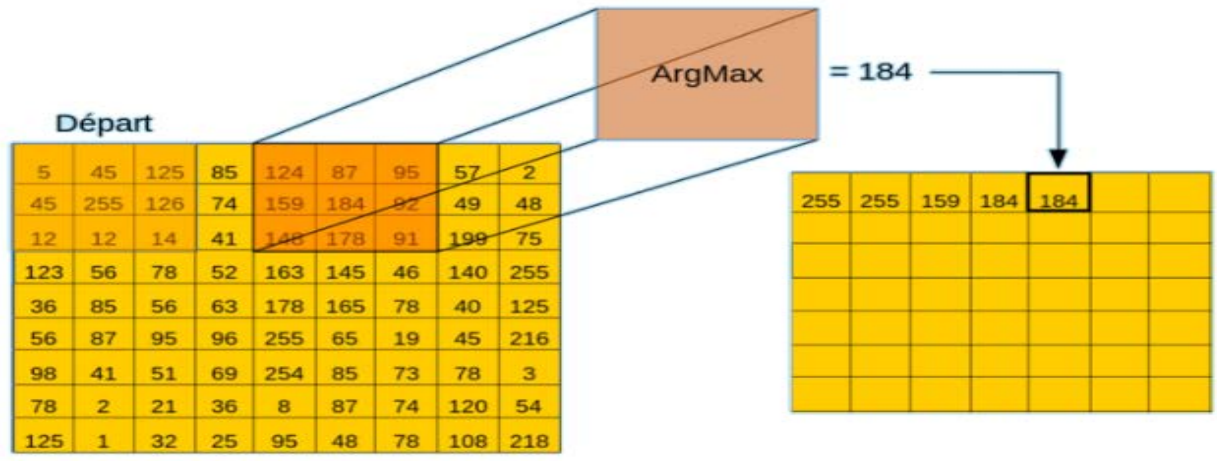


Figure 4: Exemple de La couche de MaxPooling [8].

3.3 La couche Dropout

Le dropout est une technique destinée à empêcher le sur-ajustement sur les données de training en abandonnant des unités dans un réseau de neurones. En pratique, les neurones sont soit abandonnés avec une probabilité pour garder avec une probabilité 1-p.

3.4 La couche ReLU (Rectified*LinearUnits*)

La couche ReLU désigne la fonction réelle non-linéaire définie par ReLU(x)=max (0, x) [9].

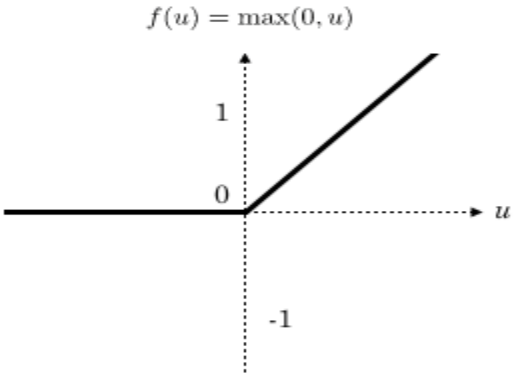


Figure 5: Allure de la fonction ReLU.

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros (figure 5). Elle joue le rôle de fonction d'activation.

3.5 La couche complètement connectée

La couche complètement connectée (en anglais fully connected layer) (FC) s'applique sur une entrée préalablement aplatie où chaque entrée est connectée à tous les neurones (figure 6). Les couches FC sont typiquement présentes à la fin des architectures de CNN et peuvent être utilisées pour optimiser des objectifs tels que les scores de classe.

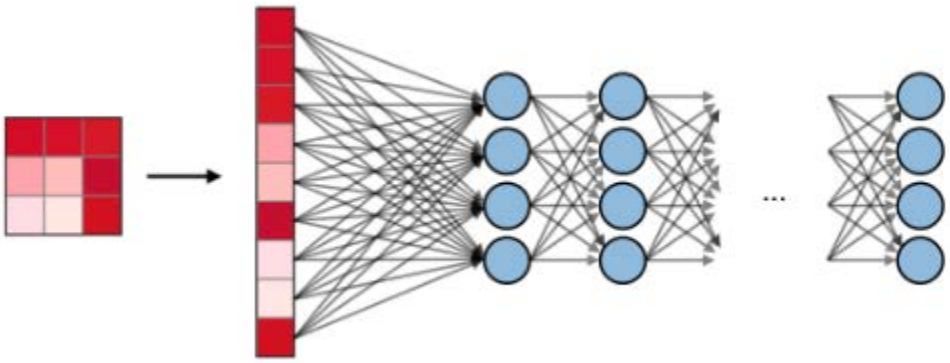


Figure 6: La couche FullyConnected [10].

3.6 Couche de perte (LOSS)

La couche de perte 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 » permet de calculer la distribution des probabilités d'appartenance aux classes sur la sortie.

4. Exemples de modèles de CNN

La forme la plus commune d'une architecture CNN empile quelques couches Conv-ReLU, les suit avec des couches Pool, et répète ce schéma jusqu'à ce que l'entrée soit réduite dans un espace d'une taille suffisamment petite. À un moment, il est fréquent de placer des couches entièrement connectées (FC). La dernière couche entièrement connectée est reliée vers la sortie. Voici quelques architectures communes CNN qui suivent ce modèle :

- INPUT -> CONV -> RELU -> FC
- INPUT -> [CONV -> RELU -> POOL] * 2 -> FC -> RELU -> FC Ici, il y a une couche de CONV unique entre chaque couche POOL
- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> FC Ici, il y a deux couches CONV empilées avant chaque couche POOL.

L'empilage des couches CONV avec de petits filtres de Pooling (plutôt qu'un grand filtre) permet un traitement plus puissant, avec moins de paramètres [11]. Cependant, avec l'inconvénient de demander plus de puissance de calcul (pour contenir tous les résultats intermédiaires de la couche CONV) (figure 7).

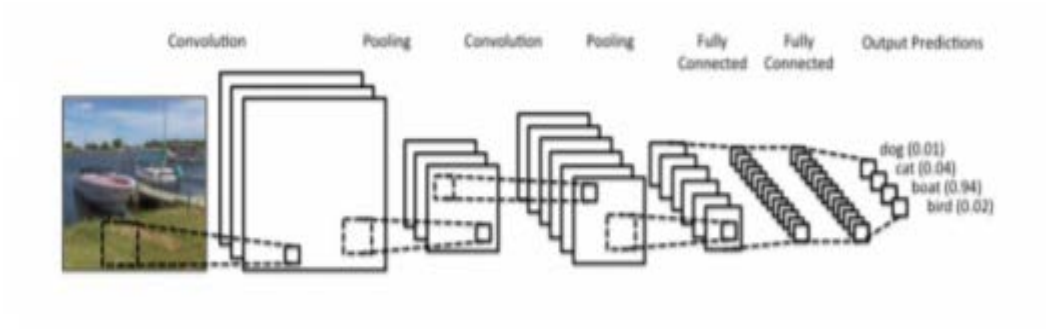


Figure 7: Exemples de modèles de CNN [11].

5. Paramètre du filtre

La couche convolutionnelle contient des filtres pour lesquels il est important de savoir comment ajuster ses paramètres.

5.1 Dimensions d'un filtre

Un filtre de taille F×F appliqué à une entrée contenant C canaux est un volume de taille F×F×C qui effectue des convolutions sur une entrée de taille I×I×C et qui produit un feature map de sortie (aussi appelé activation map) de taille O×O×C.

5.2 Nombre de filtre

Comme la taille des images intermédiaires diminue avec la profondeur du traitement, les couches proches de l'entrée ont tendance à avoir moins de filtres tandis que les couches plus proches de la sortie peuvent en avoir davantage. Pour égaliser le calcul à chaque couche, le produit du nombre de caractéristiques et le nombre de pixels traités est généralement choisi pour être à peu près constant à travers les couches. Pour préserver l'information en entrée, il faudrait maintenir le nombre de sorties intermédiaires (nombre d'images intermédiaire multiplié par le nombre de positions de pixel) pour être croissante (au sens large) d'une couche à l'autre.

5.3 Forme de filtres

Les formes de filtre varient grandement dans la littérature. Ils sont généralement choisis en fonction de l'ensemble de données. Les meilleurs résultats sur les images de la base (Modified National Institute of Standards and Technology database) MNIST [12] de taille(28x28) sont habituellement dans la gamme de 5x5 sur la première couche, tandis que les ensembles de données d'images naturelles (souvent avec des centaines de pixels dans chaque dimension) ont tendance à utiliser de plus grands filtres de première couche de 12x12, voire 15x15 [13].

Le défi est donc de trouver le bon niveau de granularité de manière à créer des abstractions à l'échelle appropriée et adaptée à chaque cas.

5.4 Forme du Max Pooling

Généralement, les valeurs typiques sont 2x2 mais de très grands volumes d'entrée peuvent justifier un Pooling 4x4 dans les premières couches. Cependant, le choix de formes plus grandes va considérablement réduire la dimension du signal, et peut entraîner la perte de trop d’information [13].

VI. Conclusion

Dans ce chapitre, nous avons présenté les notions importantes qui sont en relation avec l'apprentissage profond (Définition, Architectures…etc.).

Nous avons présenté en détails les réseaux de neurones convolutionnels qui constituent la base des modèles pré-trainés utilisés dans notre système. Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques.

Dans le prochain chapitre nous expliquerons les différentes étapes de la démarche suivie dans la conception de notre système proposée ainsi que les outils utilisés pour la réalisation de ce dernier.

Chapitre 3

Application

Introduction

Après avoir présenté les notions générales de la classification des images et les techniques de l'apprentissage profond, nous présentons, dans ce chapitre, la démarche que nous avons suivie dans la conception et l'application de notre système proposé.

I. Outils de développement

1. Python

Python est un langage de programmation interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du coût de la maintenance des codes.

2. Google Colab

Colab est un environnement de développement Python qui s'exécute dans le navigateur WEB. Colab offre gratuitement un niveau minimum de capacité de calcul et de stockage sans configuration requise.

3. Tensorflow

TensorFlow est un Framework de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des Frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones.

Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow : Gmail, Google Photos, Reconnaissance de voix.

4. Keras

Keras est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow ou Theano. Elle a été développée en mettant l'accent sur l'expérimentation rapide. Elle permet d'aller de l'idée aux résultats avec le moins de délai possible ce qui représente une aide considérable à la recherche scientifique.

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow.

II. Base d’images

CIFAR-10 est un sous-ensemble d’images étiquetées parmi 80 millions d'images. Il a été collecté par Alex Krizhevsky, Vinod Nair et Geoffrey Hinton dans le but de l’utiliser dans le domaine de la reconnaissance d’objet [14].

CIFAR-10 se compose de 60000 images couleur de taille 32x32, réparties en 10 classes de 6000 images chacune (figure 8).

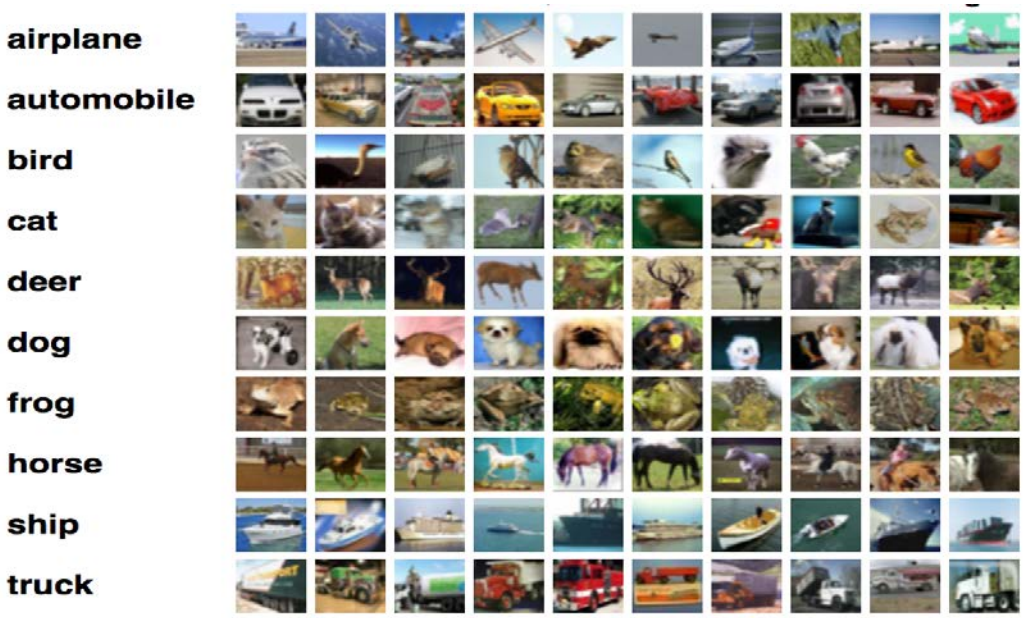


Figure 8: Base d’image CIFAR-10[14].

III. Architecture du système proposé

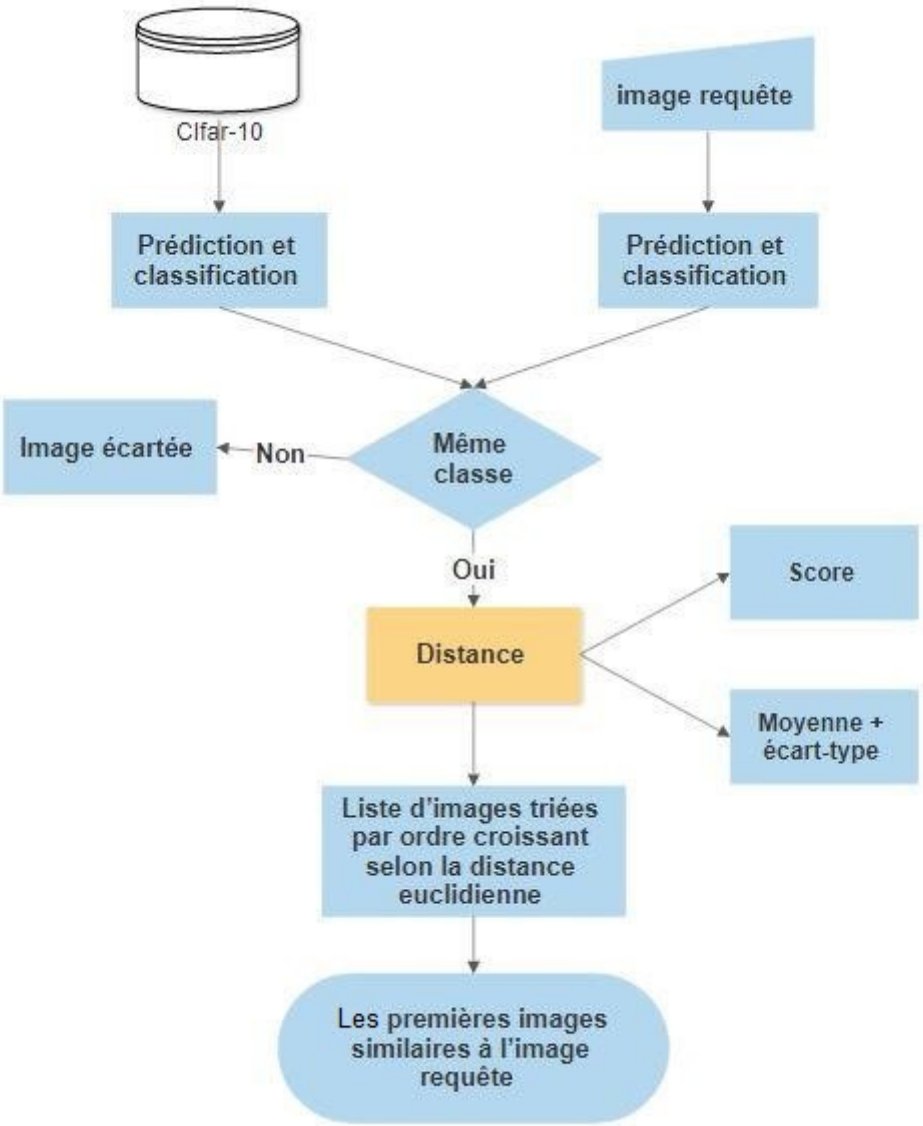


Figure 9: Architecture du système proposé.

La figure 9 schématise l'architecture de notre système de recherche des images similaires à une image requête que nous avons appliqué sur la base d'image CIFAR-10.

Notre système consiste à combiner une phase de classification avec une phase de recherche par le contenu.

Phase de classification : est une classification à dix classes basées sur les modèles pré-trainés des CNNs qu'est appliquée sur la base d'images ainsi que l'image requête. Ensuite, on ne retient que les images de même classe que la classe d'image requête.

Phase de recherche par le contenu : Consiste à calculer la distance euclidienne entre les images de la base retenues et l'image requête. Ladite distance est calculée à partir des caractéristiques des images que nous avons choisies, à savoir le score (probabilité d'appartenance d'une image à une classe attribuée par le CNN), la moyenne de l'image et son écart-type. Nous avons choisi de mesurer la similarité entre les images de la base et l'image requête en calculant la distance euclidienne selon deux modes : (i) le score uniquement et (ii) la moyenne de l'image et son l'écart type. Finalement, la liste d'images est triée par ordre croissant selon la valeur de la distance et les premières images seront retournées comme réponse à la l'image requête.

IV. Les modèles pré-trainés

Dans notre travail, nous avons utilisé quatre modèles pré-traînés de CNN à savoir ; VGG16, VGG19, ResNet50, DensNet121 (figure 10). Ce choix est dû au fait que ces derniers ont été conçus pour des images de taille 32x32 qu'est justement la taille des images de notre base d'images en l'occurrence CIFAR-10.

Les modèles que nous avons choisis sont pré-trainés sur un ensemble de données à grande échelle appelé ImageNet dont la plupart des classes sont des animaux et des objets quotidiens.

```
tf.keras.applications.Model_name(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",)
```

Figure 10: La fonction générale des modèles keras pré-trainés [15].

1. Réseau VGG

VGGNet (Visual Geometry Group Net) a été le 1er finaliste de l'ILSVRC 2014 (ImageNet Large Scale Visual Recognition Challenge) [16] dans la tâche de classement. C'est le premier réseau qui utilisait le bloc comme composant unitaire. Chaque bloc se compose d'une série de couches convolutives, suivies d'une couche de MaxPooling pour le sous-échantillonnage spatial. Les auteurs de VGGNet ont utilisé des noyaux 3x3 pour la convolution.

Tous les modèles VGG courants sont illustrés ci-dessous (figure 11) :

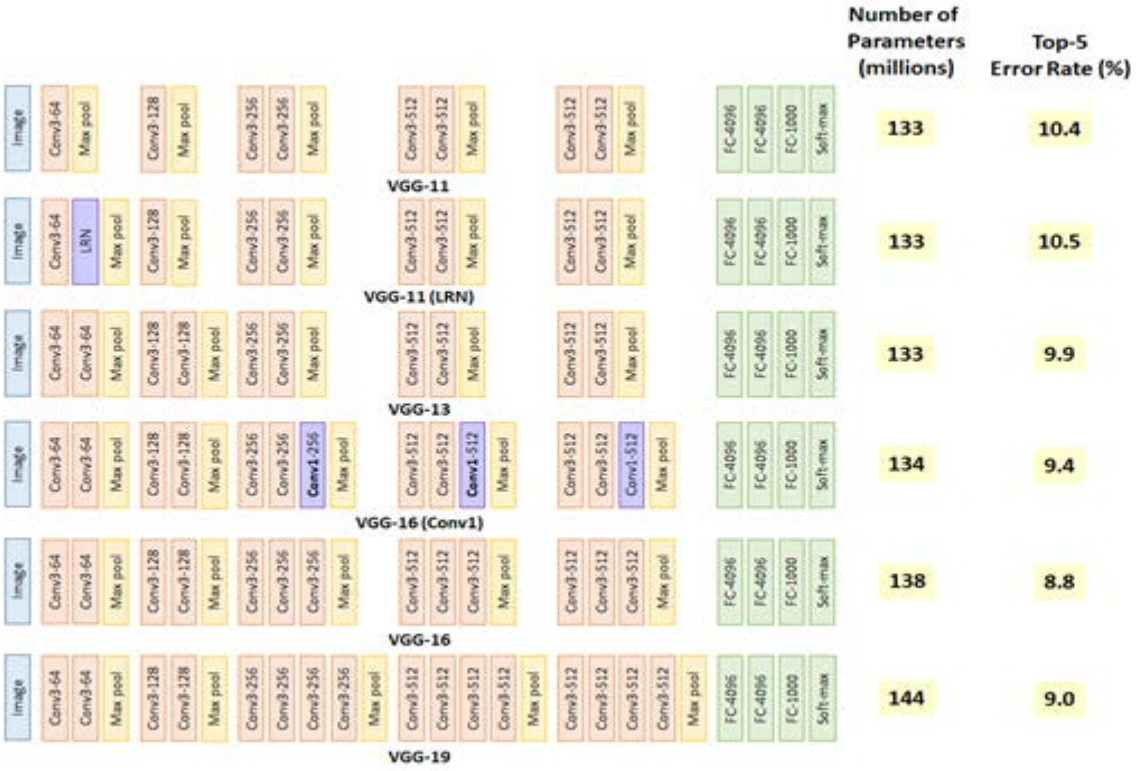


Figure 11: Architecture de Réseau VGG [17].

1.1. VGG16

Le VGG-16 est l'un des modèles pré-traînés les plus populaires pour la classification d'images. Présenté lors de la célèbre Conférence ILSVRC 2014 [16]. Développé au Visual Graphics Group de l'Université d'Oxford, VGG-16 a battu le standard d'AlexNet de l'époque et a été rapidement adopté par les chercheurs et l'industrie pour ses tâches de classification d'images.

Explorons ses couches en détail :

La chose unique à propos de VGG16 est qu'au lieu d'avoir un grand nombre d'hyper-paramètres, ses concepteurs se sont concentrés sur des couches de convolution de filtre 3x3 avec un pas (stride) 1 et ont toujours utilisé le même rembourrage (Padding) et une couche MaxPooling de filtre 2x2 de pas 2. Ils ont suivi cet arrangement de la convolution et les MaxPooling layers de manière cohérente dans toute l'architecture. Au final, le VGG16 dispose de 2 FC (couches entièrement connectées) suivis d'un Softmax pour la sortie. Le 16 dans VGG16 fait référence aux 16 couches ayant des poids. Ce réseau est un assez grand réseau et il a environ 138 millions paramètres ce qui en fait un modèle plus lent et beaucoup plus grand à entraîner que les autres [18].

1.2. VGG19

Le modèle VGG19 est une variante du modèle VGG qui en bref se compose de 19 couches. Par rapport aux réseaux de neurones convolutifs traditionnels, il a été amélioré en profondeur. Il utilise une structure alternée de plusieurs couches convolutionnelles et de couches d'activation non linéaires, ce qui est mieux qu'une seule convolution [19].

2. Réseau ResNet

ResNet, abréviation de Residual Network, est un type spécifique de réseau neuronal qui a été introduit en 2016 par Kaiming et al.

Les modèles ResNet ont connu un grand succès et remporté la 1ère place au concours de classification ILSVRC 2015 [20].

Voici l'architecture de la première variante (figure 12) : ResNet34 (ResNet50 suit également une technique similaire avec juste plus de couches).

Utilise une architecture de réseau simple à 34 couches inspirée de VGG-19 dans laquelle la connexion de raccourci est ensuite ajoutée. Ces connexions de raccourci convertissent ensuite l'architecture en réseau résiduel [21].

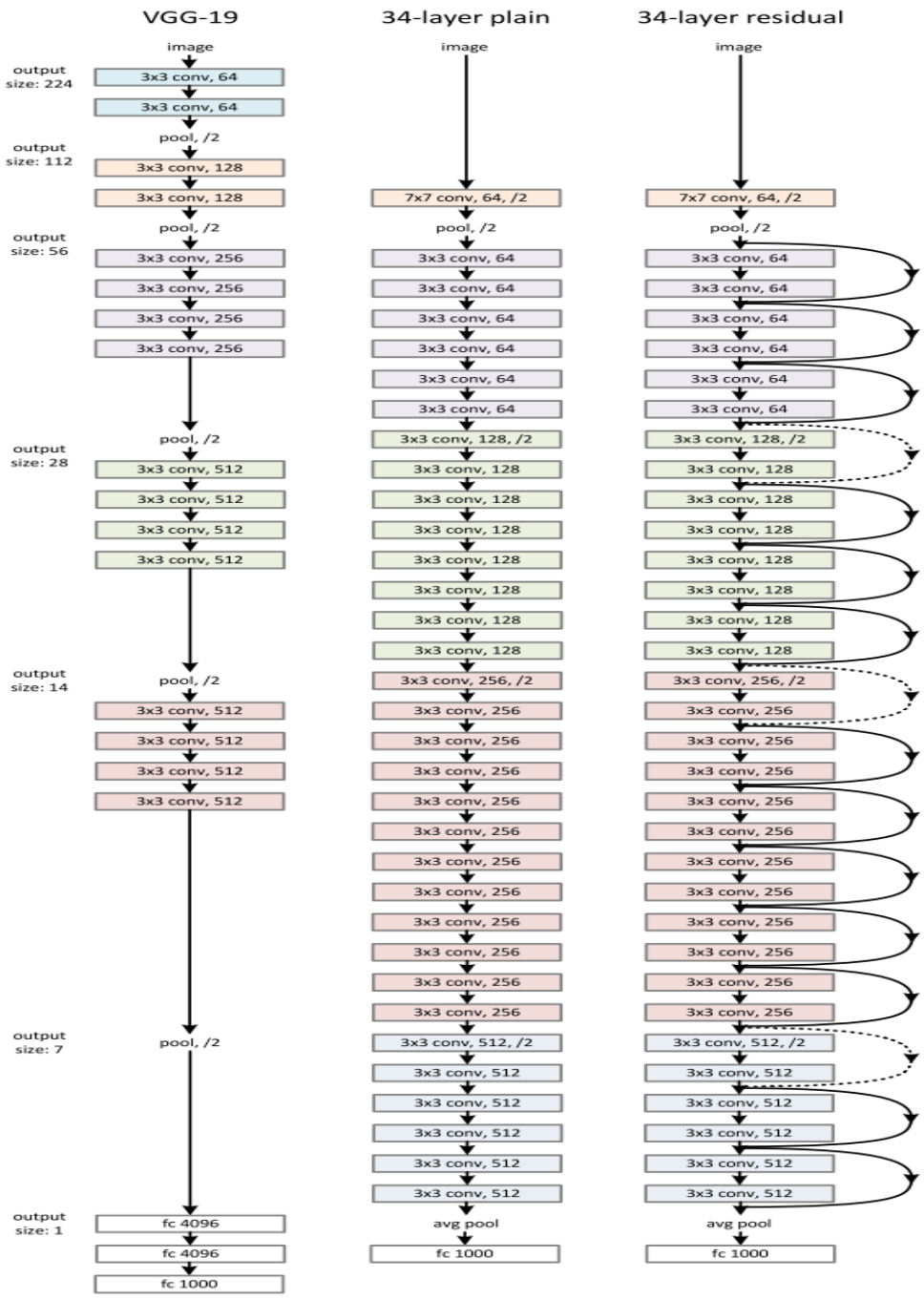


Figure 12: Architecture de Réseau ResNet [17].

2.1. ResNet-50

ResNet50 n'est pas le premier modèle issu de la famille ResNet. Le modèle original s'appelait Residual net.

La motivation principale derrière ce modèle était d'éviter une précision médiocre à mesure que le modèle devenait plus profond.

Chaque bloc ResNet a deux layers deep (utilisées dans de petits réseaux comme ResNet 18, 34) ou trois layers deep (ResNet 50, 101, 152) [22].

3. DenseNet-121

En raison du nombre très dense de connexions sur les DenseNets (Dense Convolutional Network), la visualisation devienne un peu plus complexe qu'elle ne l'était pour VGG et ResNets. La figure ci-dessous (figure 13) montre un schéma très simple de l'architecture du DenseNet-121 sur lequel nous nous concentrerons au cours de ce travail. En effet, il s'agit du simple DenseNet parmi ceux conçus sur l'ensemble de données ImageNet.

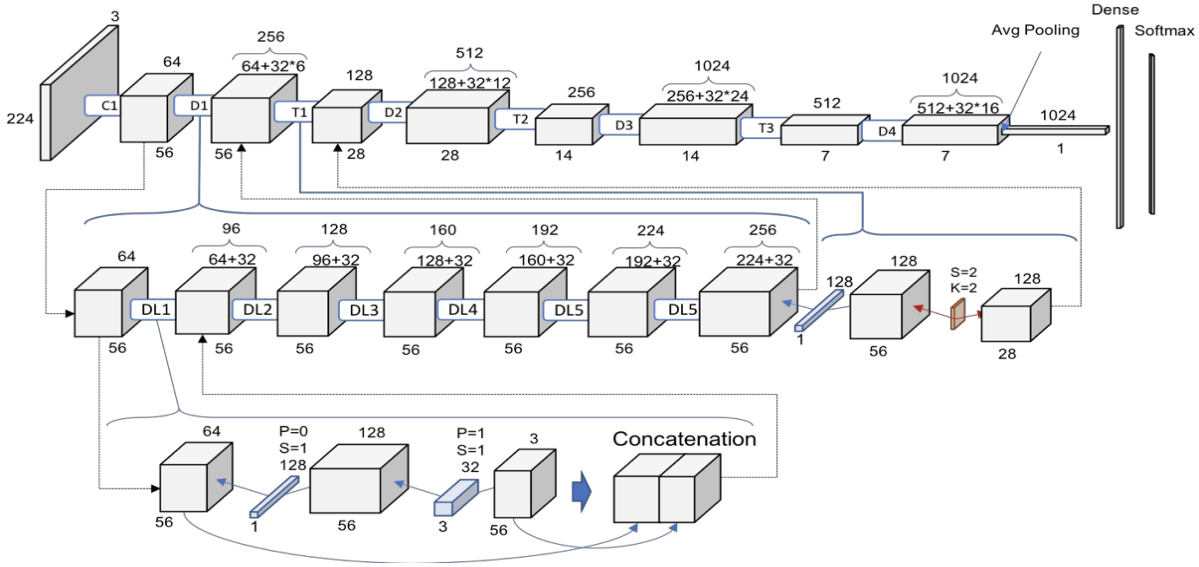


Figure 13: Un schéma sur l'architecture du DenseNet-121 [23].

V. Distance euclidienne

Nous avons choisi de mesurer la similarité entre l'image requête et les images de la base par la distance euclidienne qu'est très utilisée dans la littérature.

Aussi appelée la distance à vol d'oiseau, la distance euclidienne est une distance géométrique dans un espace multidimensionnel. Elle se calcule selon la formule suivante :

Distance $(x,y) = \{\Sigma_i (x_i - y_i)^2 \}^{½}$

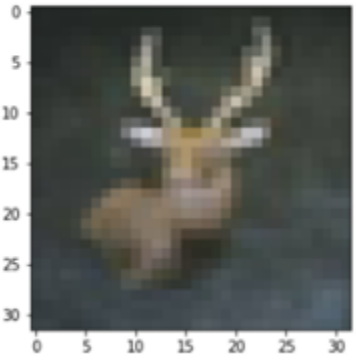
Dans notre cas, les x_i sont remplacés par les caractéristiques de l'image requête et les y_i sont remplacés par ceux des images de la base.

VI. Caractérisation des images

1. Score

Les modèles CNNs attribuent à leurs sorties des scores aux images qui représentent des probabilités d'appartenance aux classes (figure 14).

Dans ce travail, ces scores ont été prises comme caractéristiques pour les images.



```
[4.7818583e-18  1.0000000e+00
9.0201461e-27  1.8150344e-21
1.6370236e-34  9.2966842e-27
2.8366169e-26  2.0776242e-13
1.1174131e-28  3.0267504e-13]
```

Figure 14: Exemple de calcul des scores d'une image.

2. La moyenne

La moyenne d'une image est définie comme étant la moyenne des pixels dans l'image.

3. L'écart-type

L'écart-type sert à mesurer la dispersion d'un ensemble de valeurs autour de leur moyenne, écart-type des variations des intensités dans l'image.

VII. Ajustement des modèles pré-trainés

Nous avons effectué un ajustement (fine tuning) des modèles pré-trainés choisis pour pouvoir les appliquer sur notre base d'images, en l'occurrence la CIFAR-10.

1. VGG16

1.1. Modèle 1

Layer (type)	Output Shape	Param #
up_sampling2d_3 (UpSampling2	(None, 64, 64, 3)	0
vgg16 (Functional)	(None, 1, 1, 512)	14714688
flatten_3 (Flatten)	(None, 2048)	0
dense_6 (Dense)	(None, 1024)	2098176
dropout_3 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 10)	10250

```
Total params: 16,823,114
Trainable params: 9,187,850
Non-trainable params: 7,635,264
```

Figure 15: Architecture du modèle 1 du VGG16.

Nous avons gelé (freeze) les couches du VGG16 sauf les 4 dernières couches, puis nous avons ajouté Upsampling2D pour prendre plus de points de données de chaque image puis nous avons aplati (fllaten) la sortie à 1 dimension et ajouté une FC à 1024 neurones et activation ReLU puis un taux d'abandon (a dropout rate) de 0,5 et terminé par une couche « Dense » à dix neurones (10 classes) avec une activation Softmax pour la classification (figure 15).

1.2. Modèle 2

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
up_sampling2d_3 (UpSampling2 (None, 64, 64, 3)         0
_________________________________________________________________
vgg16 (Functional)           (None, None, None, 512)   14714688
_________________________________________________________________
flatten_3 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               1049088
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                5130
=================================================================
Total params: 15,768,906
Trainable params: 15,768,906
Non-trainable params: 0
```

Figure 16: Architecture du modèle 2 de VGG16.

Nous avons gardé toutes les couches entraînables (trainable), puis nous avons ajouté Upsampling2D, nous avons aplati (fllaten) la couche de sortie (output layer) à 1 dimension et ajouté une FC à 512 neurones et activation ReLU, puis un taux d'abandon (dropout rate) de seulement 0,2 et terminé par une couche « Dense » à dix neurones (10 classes) avec une activation Softmax pour la classification (figure 16).

2. VGG19

2.1. Modèle 1

Layer (type)	Output Shape	Param #
up_sampling2d_4 (UpSampling2	(None, 64, 64, 3)	0
vgg19 (Functional)	(None, 1, 1, 512)	20024384
flatten_4 (Flatten)	(None, 2048)	0
batch_normalization_3 (Batch	(None, 2048)	8192
dense_17 (Dense)	(None, 128)	262272
dropout_5 (Dropout)	(None, 128)	0
batch_normalization_4 (Batch	(None, 128)	512
dense_18 (Dense)	(None, 64)	8256
dropout_6 (Dropout)	(None, 64)	0
batch_normalization_5 (Batch	(None, 64)	256
dense_19 (Dense)	(None, 10)	650

Total params: 20,304,522
Trainable params: 20,300,042
Non-trainable params: 4,480

Figure 17: Architecture du modèle 1 de VGG19.

Dans ce modèle, nous avons gelé (freeze) toutes les pre-trained layers, puis nous avons ajouté la couche GlobalAveragePooling2D (qui calcule la valeur moyenne de chaque patch sur la carte des caractéristiques). Ensuite, nous avons ajouté deux FC à 1024 neurones et l'activation ReLU avec un taux d'abandon (dropout rate) de 0,3 et terminé par une couche « Dense » à dix neurones (10 classes) avec une activation Softmax pour la classification (figure 17).

2.2. Modèle 2

Layer (type)	Output Shape	Param #
up_sampling2d_1 (UpSampling2	(None, 64, 64, 3)	0
vgg19 (Functional)	(None, 1, 1, 512)	20024384
conv2d_2 (Conv2D)	(None, 2, 2, 64)	294976
conv2d_3 (Conv2D)	(None, 2, 2, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 1, 1, 64)	0
flatten_1 (Flatten)	(None, 64)	0
dense_5 (Dense)	(None, 512)	33280
dropout_2 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 10)	5130

Total params: 20,394,698
Trainable params: 20,394,698
Non-trainable params: 0

Figure 18: Architecture du modèle 2 de VGG19.

Nous avons gardé toutes les couches entraînables et ajouté Upsampling2D, puis ajouté un bloc à deux couches de convolution ‘Conv2D ‘ avec de petits filtres 3×3 suivis d'une couche de MaxPooling où le nombre de filtres est de 64, le Padding est utilisé sur les couches convolutives pour assurer que les formes de hauteur et de largeur de la carte de sortie des caractéristiques « output featuremaps » correspondent aux inputs. La couche utilisera la fonction d'activation ReLU. Ensuite, nous avons aplati (fllaten) la couche de sortie à une dimension et ajouté une FC à 512 neurones et l’activation ReLU, puis un taux d’abandon (learning rate) de seulement 0,2 et terminé par une couche « Dense » à dix neurones (10 classes) avec une activation Softmax pour la classification (figure 18).

3. ResNet-50

3.1. Modèle 1

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
up_sampling2d_6 (UpSampling2 (None, 64, 64, 3)         0
_________________________________________________________________
resnet50 (Functional)        (None, 1, 1, 2048)        23587712
_________________________________________________________________
flatten_7 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_15 (Dense)             (None, 512)               4194816
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_16 (Dense)             (None, 10)                5130
=================================================================
Total params: 27,787,658
Trainable params: 27,734,538
Non-trainable params: 53,120
_________________________________________________________________
```

Figure 19: Architecture du modèle 1 de ResNet-50.

Nous avons commencé par geler (freeze) les couches du modèle ResNet-50 sauf les 10 dernières couches, et ajouté Upsampling2D, puis nous avons aplati (fllaten) la couche de sortie à une dimension et ajouté une FC à 512 neurones. A la fin, une couche « Dense » à dix neurones (10 classes) avec une activation Softmax est ajoutée à la sortie (figure 19).

3.2. Modèle 2

Layer (type)	Output Shape	Param #
up_sampling2d_7 (UpSampling2	(None, 64, 64, 3)	0
up_sampling2d_8 (UpSampling2	(None, 128, 128, 3)	0
up_sampling2d_9 (UpSampling2	(None, 256, 256, 3)	0
resnet50 (Functional)	(None, None, None, 2048)	23587712
flatten_8 (Flatten)	(None, 131072)	0
batch_normalization (BatchNo	(None, 131072)	524288
dense_17 (Dense)	(None, 128)	16777344
dropout_9 (Dropout)	(None, 128)	0
batch_normalization_1 (Batch	(None, 128)	512
dense_18 (Dense)	(None, 64)	8256
dropout_10 (Dropout)	(None, 64)	0
batch_normalization_2 (Batch	(None, 64)	256
dense_19 (Dense)	(None, 10)	650

Total params: 40,899,018
Trainable params: 40,583,370
Non-trainable params: 315,648

Figure 20: Architecture du modèle 2 de ResNet-50.

Dans ce modèle, nous avons gardé toutes les couches entraînables et ajouté trois couches de Upsampling2D. Ensuite, nous avons aplati (fllaten) la couche de sortie à une dimension et ajouté une couche BatchNormalization (utilisée pour rendre le réseau plus rapide et plus stable). Ensuite, nous avons ajouté deux blocs de couches chacun est constitué d'une couche BatchNormalization, une FC et une couche Dropout de 0,5. La FC du premier bloc est à 128 neurones et du deuxième bloc est à 64 neurones. Le modèle est finalisé par une couche « Dense » à dix neurones avec une activation Softmax. (Figure 20).

4. DenseNET121

4.1. Modèle 1

```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
densenet121 (Functional)     (None, 1, 1, 1024)        7037504
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
_________________________________________________________________
dense_3 (Dense)              (None, 256)               262400
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dense_4 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 7,339,210
Trainable params: 2,457,226
Non-trainable params: 4,881,984
_________________________________________________________________
```

Figure 21: Architecture du modèle 1 de DenseNet121.

Concernant ce modèle, les couches ont été laissés figés sauf les cinquièmes couches de convolution ‘Conv5’ ont été entraînées. Ensuite, nous avons aplati (fllaten) la couche de sortie à une dimension. Nous avons ajouté deux blocs de couches chacun est constitué d’une couche BatchNormalization (utilisée pour rendre le réseau plus rapide et plus stable), une FC et une couche Dropout de 0,5. La FC du premier bloc est à 256 neurones et du deuxième bloc est à 128 neurones. Le modèle est finalisé par une couche « Dense » à dix neurones avec une activation Softmax (figure 21).

4.2. Modèle 2

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
densenet121 (Functional)     (None, 1, 1, 1024)        7037504
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 1024)              4096
_________________________________________________________________
dense (Dense)                (None, 256)               262400
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 7,339,210
Trainable params: 7,148,426
Non-trainable params: 190,784
_________________________________________________________________
```

Figure 22: Architecture du modèle 2 de DenseNet121.

Dans ce modèle (figure 22), les couches ont été laissés figés sauf les 20 dernières couches ont été entraînées. Ensuite, nous avons aplati (fllaten) la couche de sortie à une dimension. Nous avons ajouté deux blocs de couches chacun est constitué d'une couche BatchNormalization (utilisée pour rendre le réseau plus rapide et plus stable), une FC et une couche Dropout de 0,5. La FC du premier bloc est à 256 neurones et du deuxième bloc est à 128 neurones. Le modèle est finalisé par une couche « Dense » à dix neurones avec une activation Softmax.

VII. Résultats et discussion

1. Phase de classification

Nous avons divisé la base d'images CIFAR-10 en deux ensembles. Un ensemble d'apprentissage constitué de 90% des images de la base (soit 50000 images) pour entrainer nos modèles CNNs proposés. Un deuxième ensemble de test constitué des 10% restante des images de la base (soit 10000 images). Ce dernier ensemble est utilisé pour tester les modèles CNNs proposés ainsi que le système de recherche proposé.

1.1. VGG16

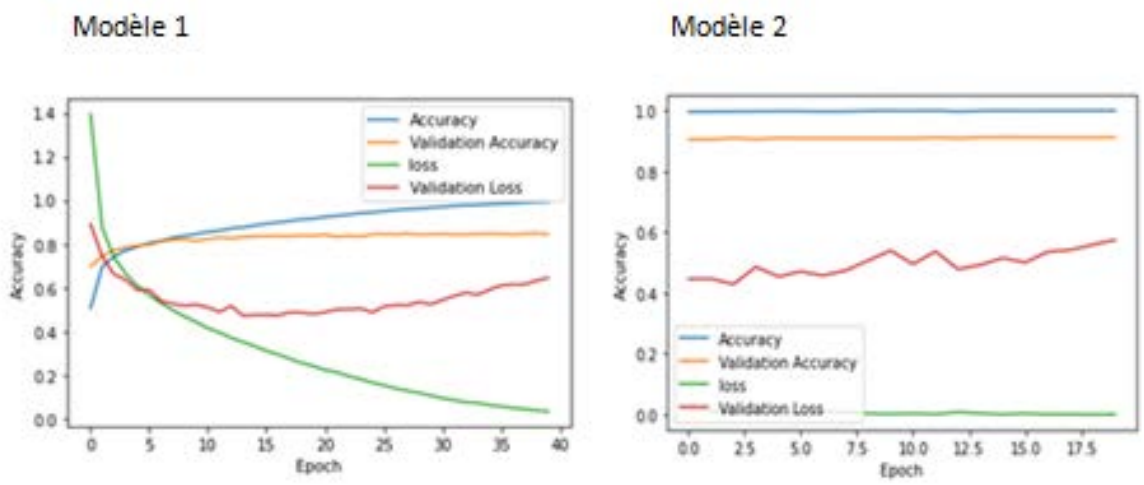


Figure 23: Graphe de précision et perte de VGG16 (modèle 1 et 2).

La figure 23 montre que la précision de validation (test accuracy) est d'environ 83,9% pour le 1er modèle et 90,3% pour le 2ème modèle. Cependant, un sur-apprentissage important (overfitting) existe toujours car la précision d'apprentissage (training accuracy) est toujours supérieure à la précision de validation et la valeur de de perte d'apprentissage (training loss) est inférieure à la valeur de perte de validation (validation loss) car la taille de notre dataset n'est pas si grande.

1.2. VGG19

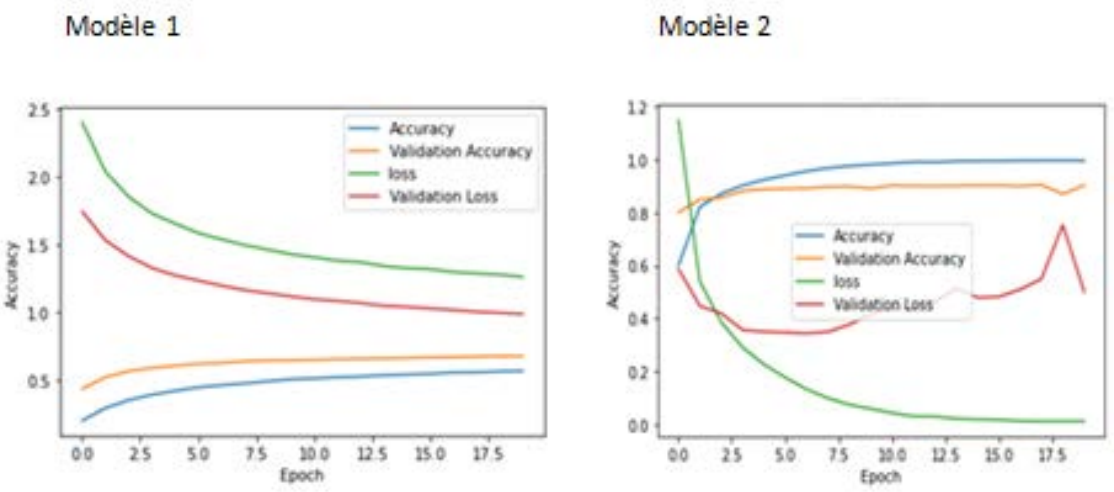


Figure 24: Graphe de précision et perte de VGG19 (modèle 1 et 2).

La figure 24 montre un test accuracy d'environ 68,3% pour le 1er modèle et 90,5% pour le 2ème modèle. Un sur-apprentissage important existe toujours avec le 2ème modèle puisque sa précision d'apprentissage est toujours supérieure à la précision de validation et que

la valeur de perte d'apprentissage est inférieure à la valeur de perte de validation. Contrairement au1er modèle qui ne manifeste pas de sur-apprentissage.

1.3. ResNet-50

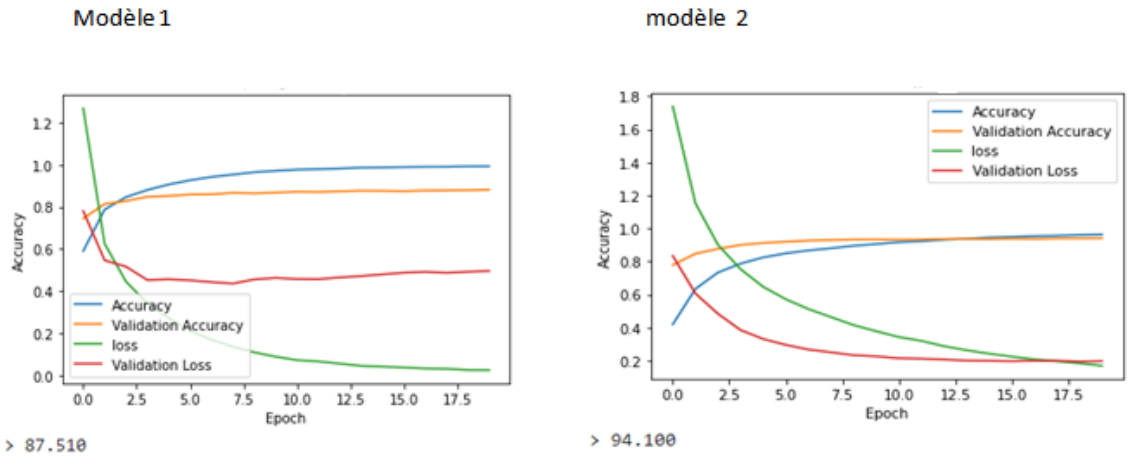


Figure 25: Graphe de précision et perte de ResNet-50 (modèle 1 et 2).

D'après la figure 25, Nous remarquons dans le graphe du 1er modèle que la précision est d'environ 99% pour la validation et 88% pour l'apprentissage. La précision d'apprentissage est bien supérieure à la précision de validation et la valeur de perte est bien inférieure à la valeur de perte de validation. Il est évident qu'il existe un fort overfitting. Cependant, la précision du deuxième modèle est 96% pour la validation et 94% pour l'apprentissage. Il est à remarquer qu'au démarrage, la précision de validation est supérieure à précision d'apprentissage et par la suite cette dernière devienne un peu meilleure que celle de la validation. De même, au démarrage la valeur de perte d'apprentissage est supérieure à celle de la validation, ensuite la valeur perte d'apprentissage devienne moins meilleure que celle de la validation. Même s'il y a un peu d'overfitting à la fin, le modèle peut être considéré comme performant. Le modèle 2 quant à lui fonctionne mieux que le modèle 1 avec une précision supérieure et une moindre perte.

1.4. DenseNet121

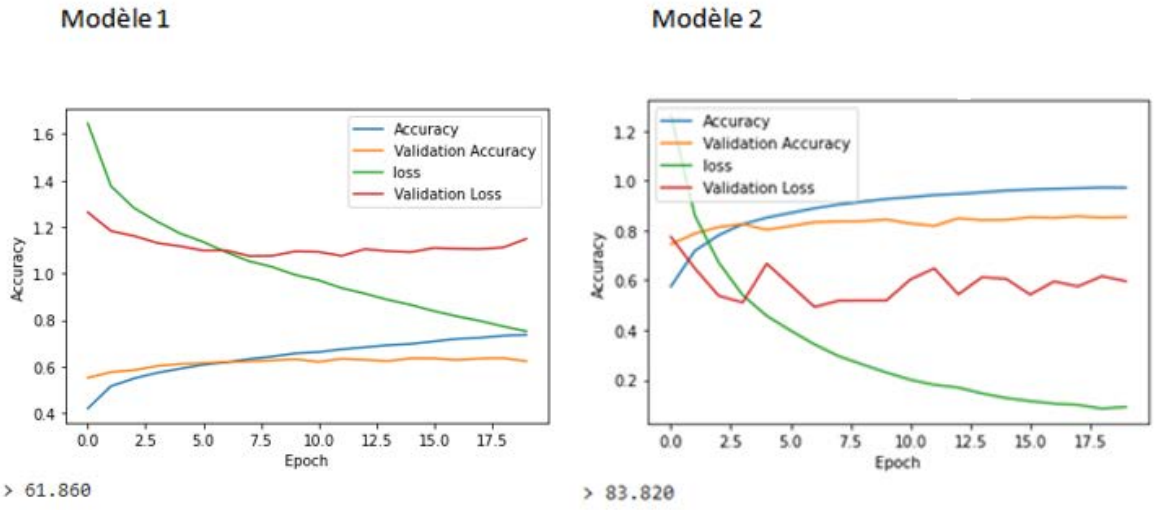


Figure 26: Graphe de précision et perte de DenseNet121 (modèle 1 et 2).

D’après les graphes de la figure 26, la précision de validation du 1er modèle est supérieure à celle d’apprentissage au démarrage ensuite cette dernière devienne progressivement supérieure à la précision de validation. Par contre, la perte d’apprentissage termine par devenir inférieure celle de la validation. La précision de validation est de 74% et celle l’apprentissage est 62%. En revanche, le modèle 2, manifeste un fort overfitting, sa précision de validation est de 96% et celle d’apprentissage est de et 85%. La précision d’apprentissage est bien supérieure à celle de la validation et la valeur de la perte en apprentissage est bien inférieure à la valeur de la perte en validation, Même s'il y a un peu d’overfitting, la performance du modèle est acceptable.

		Accuracy	Trainable params
VGG16	Model 1	83.380	9,187,850
	Model 2	90.380	15,768,906
VGG19	Model 1	68.350	1,585,162
	Model 2	90.120	20,394,698
ResNet-50	Model 1	87.510	27,734,538
	Model 2	94.100	40,583,370
DensNet121	Model 1	61.680	2,457,226
	Model 2	83.820	7,148,426

Tableau 1: Résultats des modèles choisis.

2. Phase de recherche par le contenu

Afin de perfectionner les résultats obtenus par la phase de classification nous avons ajouté une phase de recherche par le contenu qui consiste à calculer la distance euclidienne entre les caractéristiques de chaque image de la base de test ayant la même la classification que l'image requête d'un côté, et les caractéristiques l'image requête de l'autre côté. Deux modes de caractérisations ont été adoptés(i) caractérisation par le score et (ii) caractérisation par la moyenne de l'image et son écart-type.

2.1. Avec la caractéristique score

2.1.1. VGG16

Nous avons choisi d'utiliser une image d'un camion comme image requête. Les dix premières images retournées par le modèle 2 du VGG16 sont toutes des images de camions (figure 27).

Figure 27: Résultat de recherche des images similaires à une image requête d'un camion en utilisant VGG16et la caractérisation par le score.

2.1.2. VGG19

Nous avons choisi d'utiliser une image d'un oiseau comme image requête. Les dix premières images retournées par le modèle 2 du VGG19 sont toutes des images d'oiseaux (figure 28).

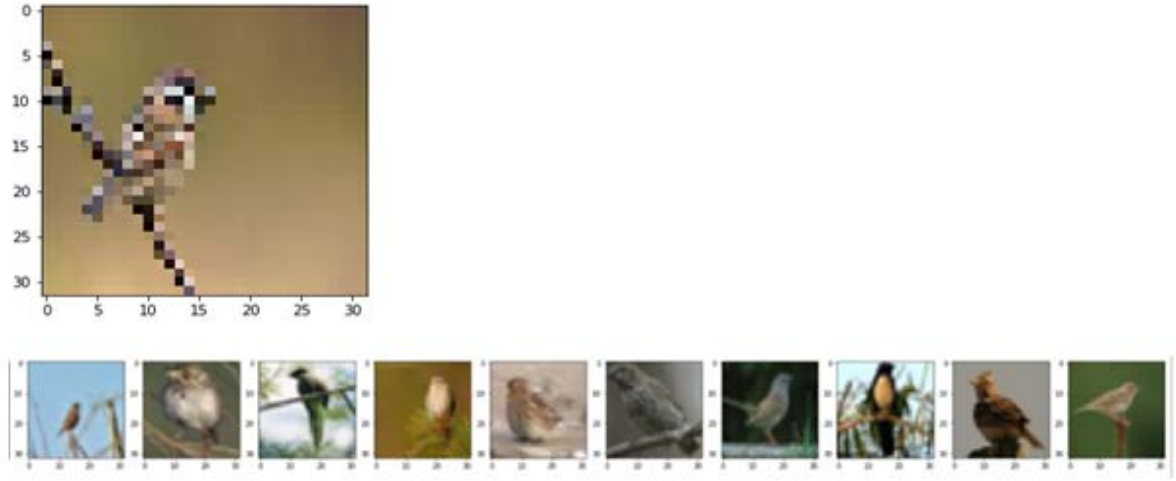


Figure 28: Résultat de recherche des images similaires à une image requête d'un oiseau en utilisant VGG19 et la caractérisation par le score.

2.1.3. ResNet-50

Nous avons choisi d'utiliser une image d'une grenouille comme image requête. Les dix premières images retournées par le modèle 2 du ResNet-50 sont toutes des images de grenouille (figure 29).

Figure 29: Résultat de recherche des images similaires à une image requête de grenouille en utilisant ResNet-50et la caractérisation par le score.

2.1.4. DensNet121

Nous avons choisi d'utiliser une image d'un bateau comme image requête. Les dix premières images retournées par le modèle 1 du DensNet121 sont toutes des images de bateaux (figure 30).

Figure 30: Résultat de recherche d'images similaires à une image requête d'un bateau en utilisant DensNet121et la caractérisation par le score.

2.2. Caractérisation par la moyenne et l'écart-type

2.2.1. GG16

Nous avons choisi d'utiliser une image d'une grenouille comme image requête. Huit images parmi les dix premières images retournées par le modèle 2 du VGG16 sont des images de grenouille (figure 31).

```
Image request prediction: frog
mean and std of the image request:
(0.6953188827614379, 0.2806351210557538)
 Dataset images prediction:
automobile
horse
dog
automobile
airplane
frog
(0.6953188827614379, 0.2806351210557538)
horse
dog
frog
(0.24087775735294117, 0.13039189148897026)
cat
deer
bird
automobile
truck
```

```
DISTANCE:
[[1.66986428e-01
 [4.46143798e-02
 [0.00000000e+00
 [1.81633680e-01
 [1.38514818e-01
 [1.70706769e-01
 [3.71511461e-02
 [1.81038858e-01
 [6.58966993e-02
 [6.68827795e-02
 [2.27944576e-01
 [1.32925426e-01
 [7.68880248e-02
 [3.99707725e-02
 [9.02535382e-02
 [9.36556169e-02
```

```
distance sorted:
[[0.00000000e+00
 [3.71511461e-02
 [3.99707725e-02
 [4.46143798e-02
 [6.58966993e-02
 [6.68827795e-02
 [7.68880248e-02
 [9.02535382e-02
 [9.36556169e-02
 [1.32925426e-01
```

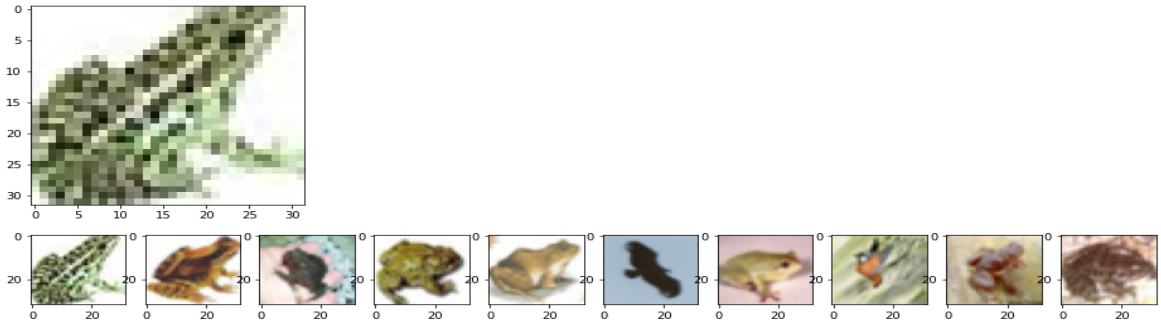


Figure 31: Résultat de recherche des images similaires à une image requête d'une grenouille en utilisant VGG16 et la caractérisation par la moyenne de l'image et son écart-type.

2.2.2. VGG19

Nous avons choisi d'utiliser une image d'un avion comme image requête. Huit images parmi les dix premières images retournées par le modèle 2 du VGG19 sont des images de grenouille (figure 32).

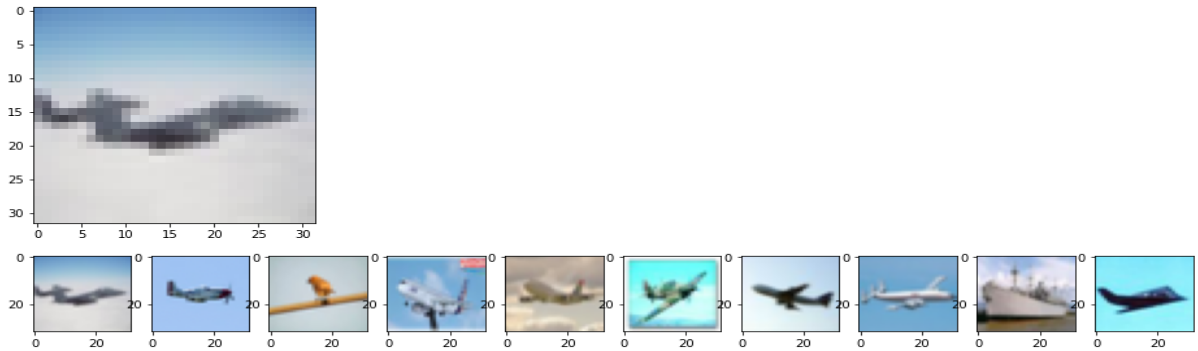


Figure 32: Résultat de recherche des images similaires à une images requête d'un avion en utilisant VGG19 et la caractérisation par la moyenne de l'image et son écart-type.

2.2.3. ResNet-50

Nous avons choisi d'utiliser une image d'un chien comme image requête. Les dix premières images retournées par le modèle 2 du ResNet-50 sont toutes des images de chiens (figure 33).

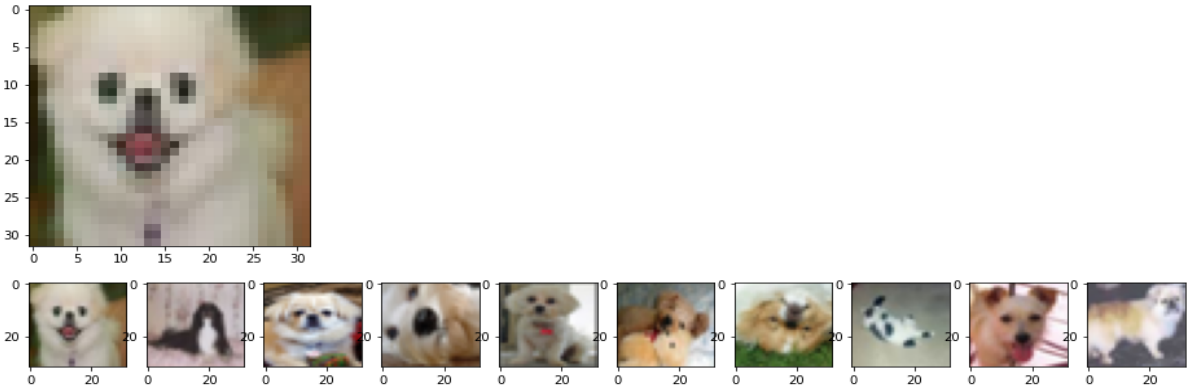


Figure 33: Résultat de recherche des images similaires à une image requête d'un chien en utilisant ResNet-50 et la caractérisation par la moyenne de l'image et son écart-type.

2.2.4. DenseNet121

Nous avons choisi d'utiliser une image d'un cheval comme image requête. Les dix premières images retournées par le modèle 1 du DensNet121 sont toutes des images de chevaux (figure 34).

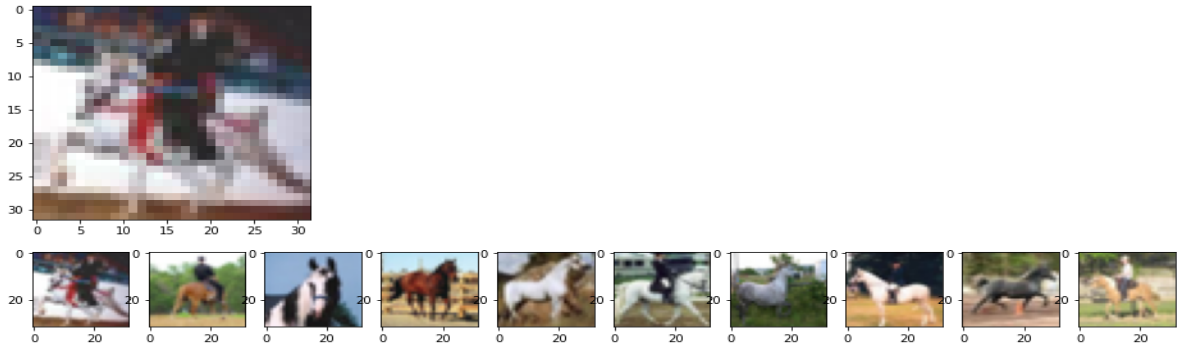


Figure 34: Résultat de recherche des images similaires à une image requête d'un cheval en utilisant DenseNet121 et la caractérisation par la moyenne de l'image et son écart-type.

Nous avons appliqué nos modèles proposés sur l'ensemble des images de test de la base CIFAR-10. Les résultats obtenus sont résumés dans le tableau 2.

		Distance (scores)	Distance (moyenne, écart-type)
VGG16	Model 1	99.9%	70.0%
	Model 2	100%	74%
VGG19	Model 1	94.99%	33%
	Model 2	99.57%	73.33%
ResNet-50	Model 1	84.99%	62.33%
	Model 2	99.9%	80%
DensNet121	Model 1	82%	23.33%
	Model 2	99.22%	55%

Tableau 2: Récapitulation des résultats de recherche d'image similaires.

D’après le tableau 2, les meilleurs résultats obtenus par nos modèles proposés sont ceux réalisés en caractérisant les images par la valeur du score : Modèle 2 de VGG16 (100%), Modèle 2 de VGG19 (99.57%), Modèle 2 du ResNet-50 (99.9%) et le modèle 2du DensNet121 (99.22%). Les résultats montrent aussi comparativement au tableau 1 que la phase de recherche a nettement amélioré les résultats de la phase de classification.

Conclusion

Dans le but de valider notre système proposé, nous avons ajusté et appliquer plusieurs modèles de CNNs pré-trainés à savoir VGG16, VGG19, ResNet-50 et DenseNet121 sur la base d’images CIFAR-10. Nous avons calculé les valeurs de score, la moyenne de l’image et son écart-type pour caractériser les images de la base. Nous avons aussi calculé la distance euclidienne entre les images de la base et les images requêtes.

Les résultats obtenus démontrent l’efficacité de notre approche et la performance de nos modèles réalisés.

Conclusion Générale

Nous avons visé dans ce travail la réalisation d'un système de recherche des images similaires à une image requête. A cet effet, nous avons proposé un système combinant d'une part une classification d'images en utilisant des modèles pré-trainés de CNNs à savoir VGG16, VGG19, ResNet-50 et DenseNet121, et de l'autre part, une mesure de similarité à savoir la distance euclidienne calculée sur la base des caractéristiques d'images (le score, la moyenne de l'image et son écart-type).

Pour ce faire, nous avons choisi d'appliquer notre système sur la base d'images Cifar-10. Plusieurs variantes de systèmes pour notre approche ont été testées en adaptant les CNNs et utilisant deux modes de caractérisation d'images (le premier mode utilise le score et le deuxième mode utilise la moyenne de l'image et son écart type).

Les résultats sont prometteurs tout en ouvrant d'autres voies de perfectionnement telles que :

- Amélioration des performances des modèles CNNs proposés.
- Exploitation d'autres modèles CNNs pré-trainés.
- Caractérisation des images par d'autres méthodes.
- Exploitation d'autres mesures de similarité.

Bibliographie

[1] Zakariyaa I., Apprentissage Supervisé Vs. Non supervisé, https://le-datascientist.fr/apprentissage-supervise-vs-non-supervise (Consultéle 15/05/2020)

[2] Borgi, A., Akdag, H. Knowledge Based Supervised Fuzzy-Classification: An Application to Image Processing. Annals of Mathematics and Artificial Intelligence 32, 67–86 (2001) (Consulté le 15/05/2020)

[3] Yann L., Recherches sur l'intelligence artificielle, https://www.college-de-france.fr/site/yann-lecun/Recherches-sur-l-intelligence-artificielle.htm (Consulté le 15/05/2020)

[4] Apprentissage profond et reseau nerenaux numerique, http://accompania.com/apprentissage-profond-et-reseau-nerenaux-numerique (Consulé le 17/05/2020)

[5] Chensi C., Feng L., Hai T., Deshou S., Wenjie S., Weizhong L., Yiming Z., Xiaochen B., Zhi X., Deep Learning and Its Applications in Biomedicine Genomics Proteomics Bioinformatics. 2018 Feb; 16(1): 17–32. (Consulté le 17/05/2020)

[6] Vartul M., Top 15 Deep Learning applications that will rule the world in 2018 and beyond, https://medium.com/breathe-publication/top-15-deep-learning-applications-that-will-rule-the-world-in-2018-and-beyond-7c6130c43b01 (Consulté le 21/05/2020)

[7] Réseau de neurones artificiels : quelles sont leurs capacités, https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/quest-ce-quun-reseau-neuronal-artificiel/ (Consulté le 21/05/2020)

[8] Les réseaux de neurones convolutifs, http://www.natural-solutions.eu/blog/la-reconnaissance-dimage-avec-les-rseaux-de-neurones-convolutifs (Consulté le 21/05/2020)

[9] Découvrez les différentes couches d'un CNN, https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn Consulté le 30/05/2020)

[10] Afshine A., Shervine A.,Pense-bête de réseaux de neurones convolutionnels, https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels (Consulté le 30/05/2020)

[11] Réseau neuronal convolutif, https://fr.wikipedia.org/wiki/Réseau_neuronal_convolutif (Consulté le 08/06/2020)

[12] The MNIST DATABASE,http://yann.lecun.com/exdb/mnist/ (Consulté le 08/06/2020)

[13] Réseau neuronal convolutif,https://fr.wikipedia.org/wiki/Réseau_neuronal_convolutif (Consulté le 08/06/2020)

[14] The CIFAR-10 dataset,https://www.cs.toronto.edu/~kriz/cifar.html (Consulté le 08/10/2020)

[15] VGG16 and VGG19,https://keras.io/api/applications/vgg/#vgg16-function (Consulté le 08/10/2020)

[16] VGG16 function, http://www.image-net.org/challenges/LSVRC/2014/results (Consulté le 12/10/2020)

[17] Doan C., CIFAR10 94% Of Accuracy By 50 Epochs With End-to-End Training ,https://blog.fpt-software.com/cifar10-94-of-accuracy-by-50-epochs-with-end-to-end-training (Consulté le 12/10/2020)

[18] Rohit T.,Step by step VGG16 implementation in Keras for beginners,https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c#:~:text=VGG16%20is%20a%20convolution%20neural,competition%20in%202014.&text=It%20follows%20this%20arrangement%20of,by%20a%20softmax%20for%20output (Consulté le 12/10/2020)

[19] CNN VGG 16 and VGG 19,http://datahacker.rs/deep-learning-vgg-16-vs-vgg-19/ (Consulté le 08/10/2020)

[20] Large Scale Visual Recognition Challenge 2015 (ILSVRC2015),http://image-net.org/challenges/LSVRC/2015/ (Consulté le 12/10/2020)

[21] Hussain M,Introduction to Resnet or Residual Network,https://www.mygreatlearning.com/blog/resnet/ (Consulté le 15/10/2020)

[22] Prakash J., Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification,https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624 (Consulté le 15/10/2020)

[23] Pablo R., DenseNet on CIFAR10,https://towardsdatascience.com/densenet-on-cifar10-d5651294a1a8 (Consulté le 15/10/2020)

[24] Kaiming H., Xiangyu Z., Shaoqing R., Jian S., Deep Residual Learning for Image Recognition, Computer Vision and Pattern Recognition (CVPR), 2016.