République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Chadli Bendjedid
Faculté des Sciences et de la Technologie
Département d'Informatique

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم الاعلام الالي

جامعة الشاذلي بن جديد
UNIVERSITE CHADLI BENDJEDID

MEMOIRE

Présenté par

KERDOUD Fateh

Pour l'obtention de diplôme de

MASTER

Filière : Informatique

Spécialité : Systèmes Informatiques Intelligents

Thème

Annotation des images par une méthode d'apprentissage profond.

Soutenu le : / / 2022 Devant le Jury composé de :

Qualité	Nom et Prénom	Grade	Université
Présidente	Mme. CHEBBAT	MAB	Chadli Bendjedid El-Tarf
Rapporteuse	**Mme. BOUGUERNE**	**MCB**	**Chadli Bendjedid El-Tarf**
Examinatrice	Mme. AHMED MALAK	MAA	Chadli Bendjedid El-Tarf

Année Universitaire : 2023/2024

Remerciements

C'est avec une immense gratitude que je tiens à remercier toutes les personnes qui ont contribué, de près ou de loin, à la réalisation et à l'aboutissement de ce travail.

Tout d'abord, je voudrais exprimer ma profonde reconnaissance envers le Tout-Puissant, Allah, qui m'a guidé et soutenu tout au long de cette entreprise.

Mes sincères remerciements s'adressent ensuite au Dr.BOUGUERNE Imen, enseignante au département des mathématiques et informatique de la faculté des sciences exactes et sciences de la nature et de la vie à l'université Chadli Bendjedid El-Tarf. Je la remercie chaleureusement pour son encadrement attentif, son assistance précieuse, son soutien constant et ses précieux conseils qui ont été déterminants dans la réussite de ce mémoire.

Je tiens également à remercier Dr. AHMED MALAK et Dr. CHEBBAT pour l'honneur qu'ils m'ont fait en acceptant d'évaluer mon travail.

Enfin, j'adresse mes plus sincères remerciements à l'ensemble de mes enseignants, qui m'ont transmis avec dévouement les connaissances et compétences nécessaires pour mener à bien mes études.

À tous, je vous exprime ma plus profonde gratitude. Que Dieu vous bénisse.

بسم الله الرحمن الرحيم

Dédicaces

Je dédie ce travail de thèse avec une profonde gratitude à ma famille bien-aimée :

À la mémoire de mon cher père Abdelkrim, que Dieu l'ait en Sa sainte miséricorde. Ton amour et tes encouragements résonnent toujours en moi.

À ma mère chérie Nouara, pour ton soutien inconditionnel et tes mots bienveillants qui m'ont toujours porté.

À ma merveilleuse épouse et à mes filles adorées Zineb, Meriem et Rokia, qui m'ont accompagné avec tant d'amour et de sacrifices.

J'adresse également mes sincères remerciements à toute ma famille, mes frères et sœurs, ainsi qu'à mes précieux amis.

Votre présence à mes côtés a été un soutien inestimable.

Que Dieu vous bénisse tous pour votre affection et votre générosité.

KERDOUD Fateh

Table des matières

Liste des figures

Liste des figures

Liste des tableaux

Introduction générale

Introduction générale

La communication orale demeure le mode d'expression prédominant au sein de la société. Cependant, certaines personnes, en particulier les sourds et les malentendants, font face à des obstacles pour s'exprimer oralement. Pour surmonter ces défis, des solutions technologiques innovantes, telles que les techniques d'apprentissage profond, permettent désormais d'annoter et de reconnaître automatiquement les gestes et signes de la langue des signes. Ces avancées facilitent grandement la communication bidirectionnelle entre les communautés sourdes ou malentendantes et le reste de la population entendante.

Le présent projet vise à traduire la langue des signes américaine (ASL) en texte en employant des techniques d'apprentissage profond, notamment les réseaux neuronaux convolutionnels (CNN), afin d'aider les personnes sourdes et malentendantes à communiquer plus aisément.

On va essayer de concevoir une architecture prometteuse consiste en une séquence de couches convolutionnelles, une telle architecture devrait permettre une annotation précise et fluide de la langue des signes américaine.

Donc quelle sera la structure CNN optimale de ce modèle pour atteindre cette finalité?

Pour répondre à cette question on a vu nécessaire d'organiser ce mémoire de la manière suivante :

Tout d'abord, nous commençons ce mémoire par une introduction afin de se mettre dans le contexte d'étude, ainsi poser la problématique traitée dans ce mémoire.

Dans le premier chapitre, nous nous concentrerons sur les l'état d'art de notre travail, notamment sur l'image et ses différents types, sur l'intelligence artificielle et sa taxonomie, nous introduirons également les principes fondamentaux de ce domaine, en mettant l'accent sur les techniques que nous exploiterons plus particulièrement dans nos développements, à savoir le deep learning ainsi qu'une exposition du domaine de traitement d'image comme étant un domaine ciblé par les algorithmes d'apprentissage profonds, nous étalerons aussi le langage de signe Américain et enfin une section consacré aux travaux antérieurs menés dans le contexte de l'intégration d'algorithmes d'apprentissage profonds pour l'annotation du langage du signe Américain.

Le chapitre suivant sera entièrement consacré aux réseaux de neurones convolutionnels (CNN) qui se sont avérés particulièrement adaptés au traitement des images. Dans ce chapitre, nous présenterons, une introduction détaillée aux réseaux de neurones convolutionnels, expliquant leur fonctionnement et leurs principales caractéristiques ainsi que toutes Les notions fondamentales nécessaires à la compréhension et à l'utilisation des CNN, ses technique de régularisation et enfin une exposition de Deux approches de Développement d'un nouveau modèle CNN ; from scratch ou par l'utilisation du transfert d'apprentissage (transfer learning) à partir de modèles préentraînés.

Le prochain volet exposera la conception globale et détaillée des deux modèles du système d'annotation du langage ASL ainsi que les algorithmes généraux y associés.

Le dernier chapitre porte sur l'implémentation et l'évaluation de nos modèles. Nous commencerons par présenter les outils essentiels à l'implémentation, puis nous décrirons leur mise en œuvre et mesurerons leurs performances à l'aide des principales métriques du deeplearning. Enfin, nous comparerons ces résultats aux travaux connexes

Enfin nous clôturons notre travail par une conclusion générale qui offrira ainsi une vue d'ensemble de nos contributions, tout en identifiant les domaines d'amélioration et en ouvrant sur de futures perspectives de recherche.

Chapitre01 :

Etat de l'Art

1-Introduction :

L'annotation des signes de la langue américaine (American Sign Language - ASL) par le deep learning est une application de l'intelligence artificielle qui vise à détecter, reconnaître et comprendre les signes spécifiques de la langue des signes américaine (ASL).
L'utilisation du deep learning dans l'annotation des signes de langue américaine permet d'exploiter les avancées en matière de réseaux de neurones profonds pour analyser les séquences d'images ou de vidéos représentant les gestes des signes. Les modèles de deep learning peuvent apprendre à extraire des caractéristiques discriminantes à partir de ces données visuelles et les utiliser pour classer et reconnaître les signes spécifiques de l'ASL.

2-L'Intelligence artificielle :

La capacité d'un ordinateur numérique ou d'un robot contrôlé par ordinateur à effectuer des tâches communément associées aux êtres intelligents. Le terme est fréquemment appliqué au projet de développement de systèmes dotés des processus intellectuels caractéristiques des êtres humains, tels que la capacité de raisonner, de découvrir des significations, de généraliser ou d'apprendre à partir d'expériences passées.[**02]**

3-L'évolution de l'intelligence artificielle:

3-1-La première période 1950-1970 :

Résolution de problèmes triviaux, aucune utilité pratique, GOFAI - Good old-fashioned AI **[05]** :

1942 Les 3 lois de la robotique par Isaac Asimov, D'autres ensembles de lois ont été proposés par des chercheurs depuis lors.
1950 Le test de Turing proposé par Alan Turing
1952 Premier programme de jeu auto-apprenant
1956 Conférence de Dartmouth, première utilisation des termes « Intelligence Artificielle/IA ».
1957 General Problem Solver (GPS) par Newell
1958 McCarthy développe le langage de programmation LISP
1959 - Le laboratoire d'IA du MIT (McCarthy et Minsky) - Le terme « Machine Learning » (apprentissage automatique) par Samuel
1961 - Premier robot industriel (Unimate) travaillant chez GM - « SAINT » le premier système expert par Slagle (MIT)
1964 « STUDENT » le premier programme d'IA qui comprend le langage naturel
1965 - « ELIZA », le premier Chatbot et système expert basé sur l'IA
1966 - « Shakey », la première locomotive et le premier robot intelligent (SRI) - « MAC HACK », le programme de jeu d'échecs de Greenblatt, MIT
1968 - « SHRDLU », un des premiers programmes informatiques de compréhension du langage naturel

1970 « WABOT-1 », le premier robot anthropomorphe (Université de Waseda)
1972 « Prolog », langage de programmation logique
1973 - « Lighthill Report » : le rapport sur les faibles progrès réalisés a provoqué le « premier hiver de l'IA », qui a entraîné une réduction du financement de la recherche sur l'IA.
1974 - « MYCIN », le premier système expert d'IA basé sur des règles pour les diagnostics médicaux - Le premier véhicule autonome, un « glisseur » mécanique (Stanford)

Tableau 1.1 : Première période d'évolution de l'IA, de la ML et de la DL 1950-1970

3-2-La deuxième période 1980-2010 :

Les chercheurs alimentent les machines avec des données étiquetées, Projets : ICOT - Japon '82, MCC - US '83, Alvey - UK '84. Les algorithmes ont commencé à apparaître comme des éléments de systèmes plus vastes. Les solutions d'IA se sont avérées utiles dans l'ensemble de l'industrie technologique, comme l'exploration de données, la robotique industrielle, etc. **[05]**

1980 - Développement et commercialisation de machines basées sur LISP - « INTERNIST-1 » Le premier système expert commercial
1986 - Une camionnette sans conducteur de Mercedes-Benz, équipée de caméras et de capteurs.
1988 - « Bayesian Network », BNs ou réseaux de croyance, inventé par Pearl - Les chatbots, « Jabberwachy » et « Cleverbot » inventés par Carpenter
1989 Le premier véhicule autonome créé par la CMU à l'aide d'un réseau neuronal
1993 « Polly », le robot guide, robotique basée sur le comportement (MIT)
1997 Deep Blue d'IBM bat Gary Kasparov aux échecs
1998 « Furby », le premier robot jouet pour enfants
1999 - « Kismet », l'IA émotionnelle (MIT AI Lab) - « AIBO », le premier robot domestique doté d'une IA, présenté par Sony
2000 « ASIMO », le robot humanoïde présenté par Honda 2002 « Roomba », l'aspirateur robot autonome présenté par i-Robot
2004 - Le premier défi pour les véhicules autonomes lancé par la DARPA - Les rovers « Spirit » et « Opportunity » de la NASA explorent la planète Mars.
2005 Moteurs de recommandation basés sur l'IA
2006 - « Machine Reading » : compréhension autonome non supervisée de textes

2007 - Base de données visuelles « ImageNet » pour la recherche sur les logiciels de reconnaissance d'objets - Lancement de « CUDA » par NVIDIA, une plateforme de calcul parallèle et une interface de programmation
2009 - Construction d'une voiture autonome par Google

Tableau 1.2 : Deuxième période de l'évolution de l'IA, de la ML et de la DL 1980-2000

3-3-La troisième période de 2010 à nos jours:

L'âge de l'apprentissage automatique. Les ordinateurs acquièrent des connaissances à partir de données, et non de l'homme. Les grandes entreprises technologiques investissent dans les applications commerciales de l'IA/ML **[05]**.

2010 - La démocratisation de l'accès aux données commence pour la reconnaissance d'images - L'IA de Narrative Science démontre sa capacité à rédiger des rapports
2011 - Apple lance « Siri » - « Watson » d'IBM gagne le concours Jeopardy Chapitre 1 Principes fondamentaux de la reconnaissance des visages 10
2013 - « NEIL » de CMU, un système ML d'analyse sémantique d'images - « Vicarious » réussit le premier test de Turing - CAPTCHA
2014 - « Cortana » de Microsoft - « Alexa » d'Amazon
2015 - « TensorFlow » par Google Brain, une bibliothèque ML (TPU) - « Open AI » initiative open source pour développer l'IA au bénéfice de toute l'humanité
2016 - « Google Home » de Google - « Alpha Go » Deepmind de Google a battu le champion N°1 de Go - NVIDIA annonce un supercalculateur pour le DL et l'IA - « Sophia » robot humanoïde de Hanson Robotics, le premier robot citoyen - « PyTorch » bibliothèque ML open source
2017 - Le laboratoire de recherche en IA de Facebook a entraîné deux chatbots à communiquer entre eux afin d'apprendre à négocier ; les chatbots se sont éloignés du langage humain et ont inventé leur propre langage pour communiquer l'un avec l'autre. - Caffe » Cadre DL open source.
2018 - « BERT » par Google, la première représentation bidirectionnelle non supervisée du langage - « Bixby » présenté par Samsung - Facebook détecte les visages et partage les photos avec les amis à qui ces photos appartiennent - L'IA de traitement du langage d'Alibaba a surpassé l'intelligence humaine lors d'un concours de Standford.
2020 - L'équipe DeepMind utilise des algorithmes de DL « Agent 57 » qui surpassent les humains aux jeux Atari avec l'apprentissage par renforcement profond. - Déploiement généralisé du réseau « 5G » dans le monde entier (2020) Open AI lance GPT-3, un modèle de traitement du langage naturel capable de produire des textes de la manière dont les humains parlent et écrivent.

- (2021) Open AI s'appuie sur GPT-3 pour créer DALL-E, qui peut générer des images à partir d'invites textuelles.
- (2022) Open AI lance Chat GPT, un chatbot alimenté par un grand modèle linguistique qui gagne rapidement plus de 100 millions d'utilisateurs.
(2023) Microsoft lance une version de Bing, son moteur de recherche, alimentée par l'IA et basée sur la même technologie que Chat GPT.
(2023) Google introduces Bard, a competing conversational AI
En 2024, l'IA continue de progresser rapidement dans divers domaines, tels que la santé, les transports autonomes et la reconnaissance du langage naturel. Les entreprises comme Microsoft et Google investissent massivement dans des technologies d'IA avancées pour améliorer leurs produits et services. Les questions éthiques et la réglementation de l'IA restent des préoccupations majeures à mesure que son utilisation se généralise

Tableau 1.3 : Troisième période de l'évolution de l'IA, de la ML et de la DL 2010-jusqu'au présent

4-Taxonomie de l'intelligence artificielle :

Le machine learning, l'apprentissage profond et les réseaux neuronaux sont tous des sous-domaines de l'intelligence artificielle. Cependant, les réseaux neuronaux sont en fait un sous-domaine de la machine learning, tandis que l'apprentissage profond est un sous-domaine des réseaux neuronaux.

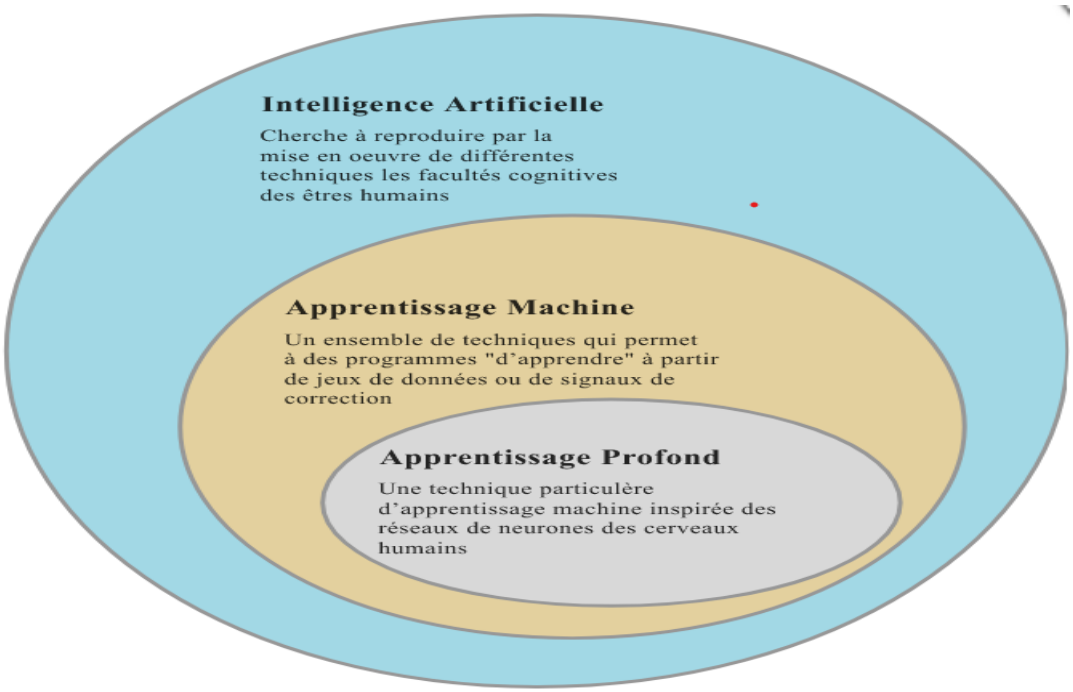


Figure1-1-Taxonomie de l'intelligence artificielle **[f02]**

4-1-L'apprentissage automatique (Machine Learning) (ML):

Les algorithmes de Machine Learning apprennent de manière autonome à effectuer une tâche ou à réaliser des prédictions à partir de données et améliorent leurs performances au fil du temps. Une fois entraîné, [03]l'algorithme pourra retrouver les patterns dans de nouvelles données.
C'est une sous-catégorie de l'intelligence artificielle sa programmation fonctionne différemment de la programmation classique .

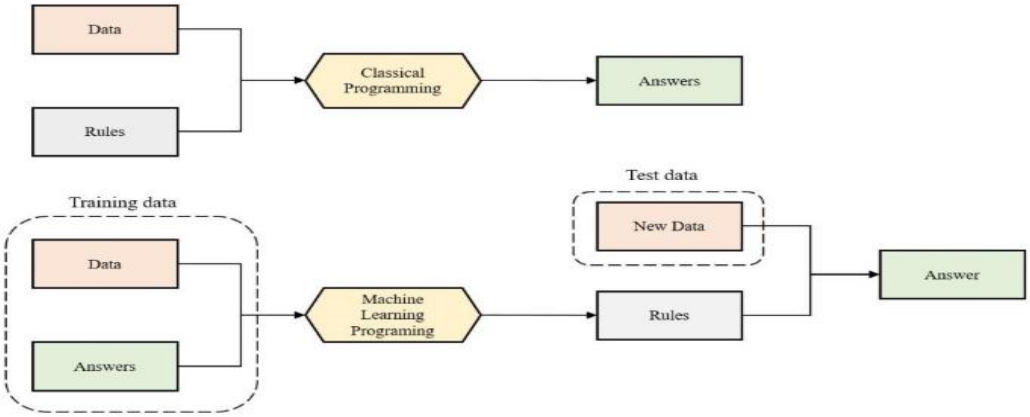


Figure1-2-Différence entre la programmation classique et l'apprentissage de la programmation ML**[f10]**

4-2--Les Réseaux de neurones profonds :

Les réseaux de neurones profonds (Deep Learning) sont une sous-catégorie de l'apprentissage automatique qui utilise des réseaux de neurones artificiels avec plusieurs couches interconnectées pour extraire des représentations hiérarchiques des données. Les réseaux de neurones profonds ont connu des avancées significatives dans des domaines tels que la vision par ordinateur, le traitement du langage naturel et la reconnaissance vocale, voici sa structure générale **[01]** :

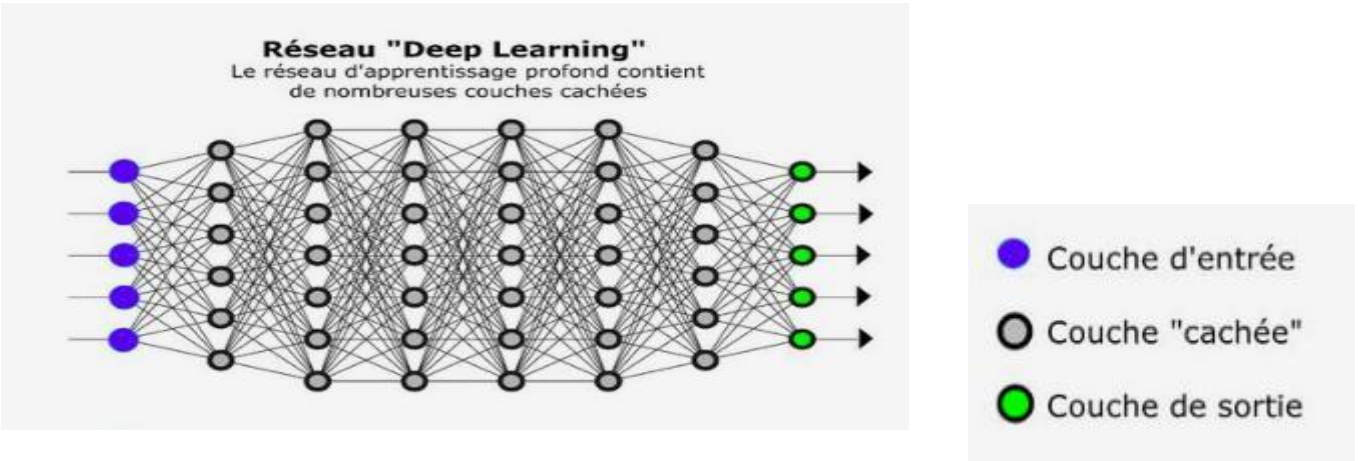


figure1-3 : structure générale des réseaux d'apprentissage profond**[f11].**

Il a été créé en s'inspirant des réseaux neuronaux qui se trouvent dans le cerveau humain. L'apprentissage profond est constitué d'un grand nombre de couches de neurones artificiels

interconnectés.

Plus le nombre de neurones est conséquent, plus le réseau est qualifié de « profond » et délivre des performances exceptionnelles**.[w05]**

5-Les Approches d'apprentissage profond :

Trois approches principales d'apprentissage automatique sont utilisées de nos jours :

5-1- L'Apprentissage supervisé:

L'apprentissage supervisé est le type d'apprentissage le plus applicable ; ses algorithmes construisent un modèle mathématique d'un ensemble de données contenant à la fois les entrées et les sorties souhaitées **[06].**

Il est défini par son utilisation d'ensembles de données étiquetés pour former des algorithmes qui permettent de classer les données ou de prédire les résultats avec précision. L'objectif est de donner un sens aux données dans le contexte d'une question spécifique. L'apprentissage supervisé est utilisé pour des problèmes de classification et de régression, comme la détermination de la catégorie à laquelle appartient un article de presse, ou la prévision du volume des ventes pour une date future donnée.

Il entraîne la machine à l'aide d'exemples x étiquetés pour avoir la relation souhaitée avec la sortie y selon la fonction **F(X)=Y.**

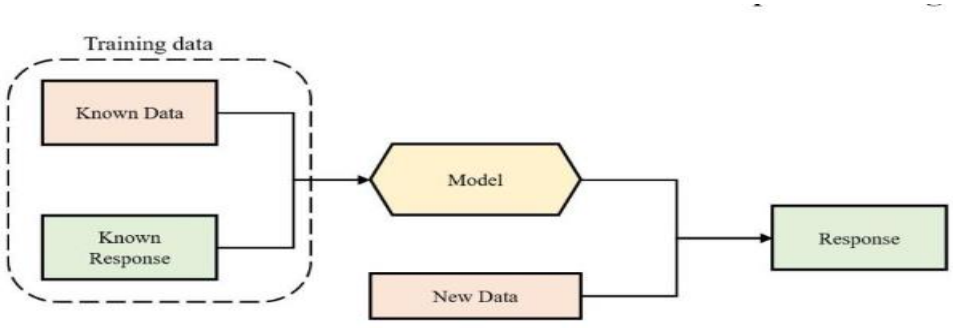


Figure1-4-fonctionnement des algorithmes basés sur l'apprentissage supervisé **[f10].**

Dans l'apprentissage supervisé, on a deux types d'algorithmes :

- Les algorithmes de régression, qui cherchent à prédire une valeur continue, une quantité.
- Les algorithmes de classification, qui cherchent à prédire une classe/catégorie.

Voici les plus essentiels :

- -Machine à vecteurs de support (SVM).
- -Réseau de neurones (RNA).
- -Méthode des k plus proches voisins (KPP).
- -Arbre de décision.
- -Classification naïve Bayesienne **[02].**

5-2- L'Apprentissage non supervisé :

Les algorithmes d'apprentissage non supervisé prennent un ensemble de données qui ne contient que des entrées, et trouvent une structure dans les données, comme le regroupement ou la mise en grappe de points de données, sans avoir préalablement déterminé une structure dans les données, comme le regroupement de points de données, sans avoir au préalable une idée de ce qu'ils peuvent être ou du nombre qu'ils devraient être, en déterminant les similitudes entre les points de données.sans intervention humaine **[f10].**

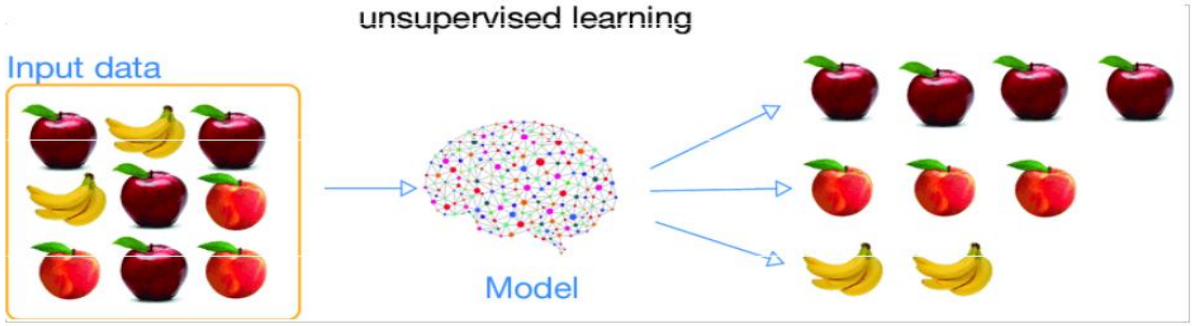


Figure 1-5-fonctionnement des algorithmes basés sur l'apprentissage non supervisé [**w03**].

Voici quelques techniques couramment utilisées en apprentissage non supervisé **[02]** :

1. Le clustering.
2. La réduction de dimension.
3. Les règles d'association.
4. La détection d'anomalies.
5. L'apprentissage génératif.

5-3- L'apprentissage par renforcement:

Il traite de l'apprentissage d'un agent intelligent à travers l'interaction avec un environnement. Il est basé sur le concept de récompense et de punition, où l'agent apprend à prendre des actions dans un environnement afin de maximiser les récompenses qu'il reçoit.
Dans l'apprentissage par renforcement, l'agent prend des décisions séquentielles en fonction de son état actuel et reçoit des récompenses ou des pénalités en réponse à ses actions. L'objectif de l'agent est de trouver une politique d'action qui maximise la récompense cumulative sur le long terme **[w3]**.

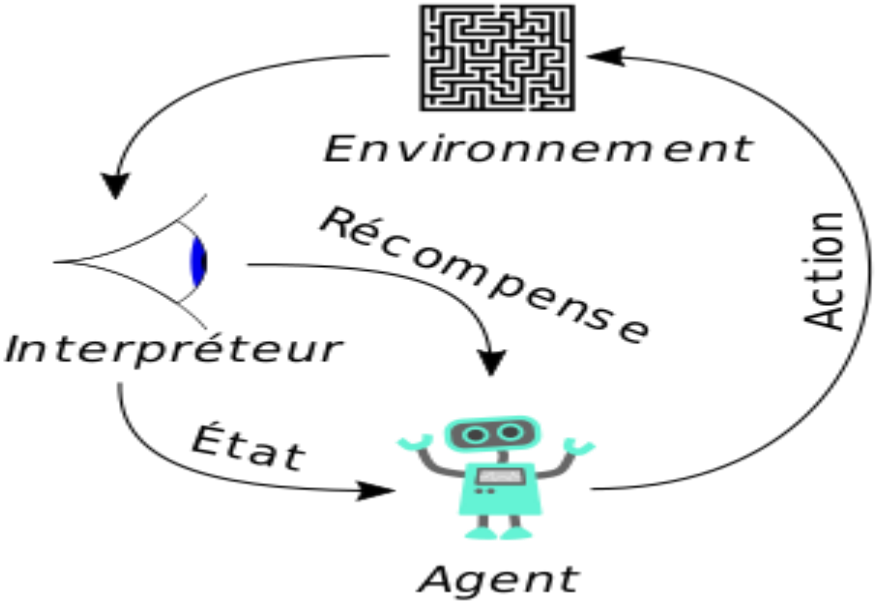


Figure1-6-fonctionnement des algorithmes basés sur l'apprentissage par renforcement [**f02**].

Certains algorithmes bien connus utilisés en apprentissage par renforcement comprennent les méthodes de

- Monte-Carlo.
- TD learning.
- Q-learning.
- le Deep Q-Network (DQN) **[03]**.

6-Quelques Algorithmes d'apprentissage profond :

Plusieurs types d'algorithmes sont utilisés en apprentissage profond**.** Chaque algorithme possède ses propres spécificités et applications on note à titre d'exemple :

6-1-Les Réseaux Adversoriels Génératifs (GAN):

Les GAN sont définis comme des algorithmes d'apprentissage profond utilisés pour générer de nouvelles instances de données qui correspondent aux données d'apprentissage. Les GAN se composent généralement de deux éléments, à savoir un générateur qui apprend à générer de fausses données et un discriminateur qui s'adapte en apprenant à partir de ces fausses données.

Les GAN fonctionnent en simulation en générant et en comprenant les données fictives et les données réelles. Au cours de l'entraînement à la compréhension de ces données, le générateur produit différents types de fausses données auxquelles le discriminateur apprend rapidement à s'adapter et à répondre comme s'il s'agissait de fausses données.

Les GAN envoient ensuite ces résultats reconnus pour qu'ils soient mis à jour. L'image ci-dessous permet de visualiser le fonctionnement. **[w06]**

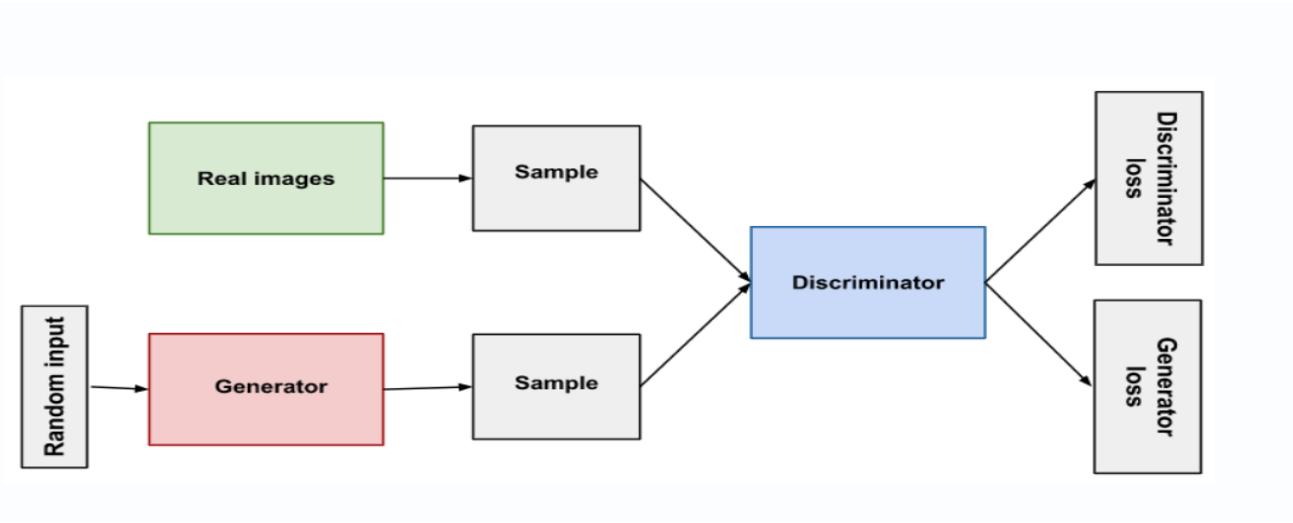


Figure1-7-fonctionnement de l'algorithme GAN **[f06]**.

6-2- Réseaux Neuronaux Récurrents (RNN):[16]

Dans les réseaux neuronaux récurrents (RNN), les sorties des états précédents servent d'entrée à l'état actuel. Les couches cachées des RNN peuvent mémoriser des informations.

L'état caché est mis à jour en fonction de la sortie générée dans l'état précédent. Le RNN peut être utilisé pour la prédiction de séries temporelles car il peut se souvenir des entrées précédentes. ce qui est appelé mémoire à long et court terme **[16].**

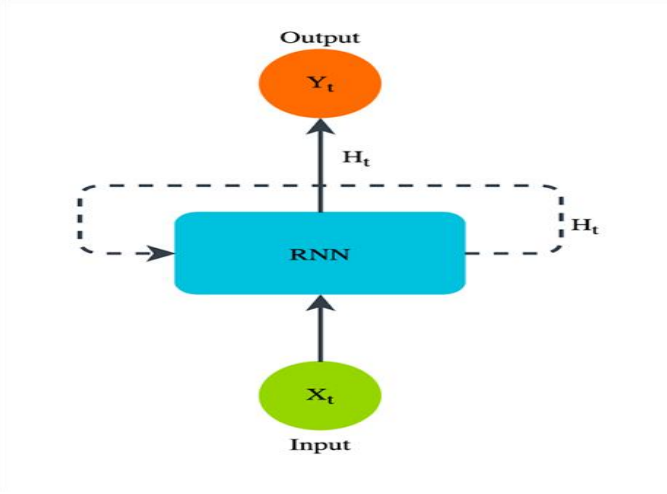


Figure 1-8- Schéma d'un réseau neuronal récurrent simple pour t étapes temporelles **[f08].**

6-3- Réseaux Neuronaux Convolutifs (CNN) :

Les réseaux neuronaux convolutifs (CNN), plus connus sous le nom de ConvNets, se composent principalement de plusieurs couches et sont spécifiquement utilisés pour le traitement d'images et la détection d'objets. Ils ont été développés en 1998 par Yann LeCun et ont d'abord été appelés LeNet. À l'époque, il a été développé pour reconnaître des chiffres et des caractères de code postal. Les CNN sont largement utilisés pour l'identification des images de satellites, le traitement d'images médicales, la prévision de séries et la détection d'anomalies. **[w06]**

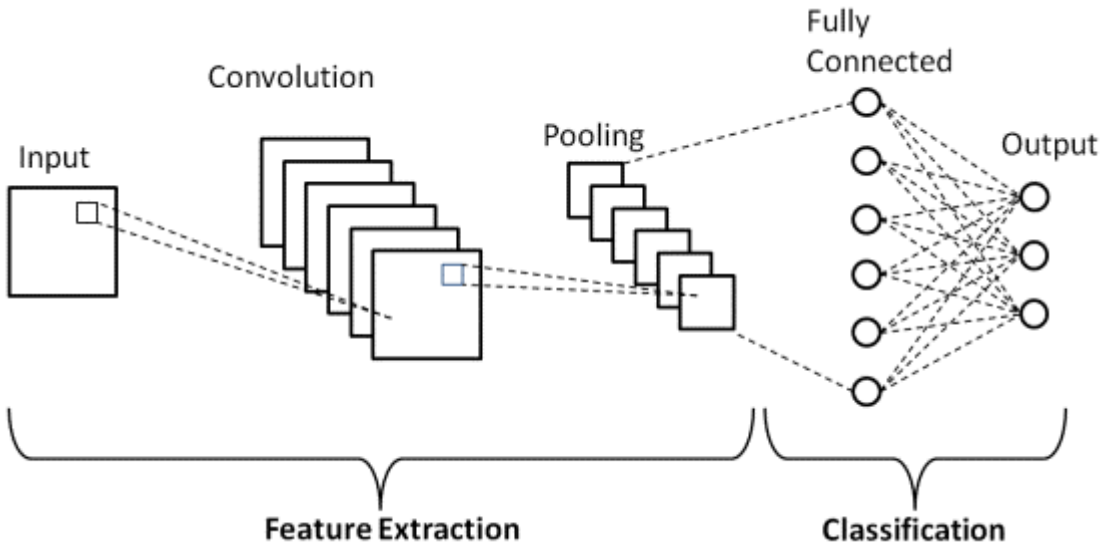


Figure 1-9-archetecture globale de l'algorithme CNN [f07].

Note :

Les CNN sont utilisés dans une variété de tâches, notamment la classification d'images, leur architecture et leur capacité à apprendre des caractéristiques visuelles font d'eux des outils puissants pour l'analyse et la compréhension des images.

Dans le chapitre suivant, on va approfondir ces concepts et explorer plus en détail l'utilisation spécifique de l'architecture CNN dans notre travail.

7- Traitement d'image :

L'apprentissage profond a eu un impact considérable sur divers domaines technologiques au cours des dernières années. L'un des sujets les plus brûlants dans ce secteur est la vision par ordinateur, c'est-à-dire la capacité des ordinateurs à comprendre les images et les vidéos par eux-mêmes. Les voitures autonomes, la biométrie et la reconnaissance faciale s'appuient toutes sur la vision par ordinateur pour fonctionner. Le traitement de l'image est au cœur de la vision par ordinateur. Le traitement d'images consiste à transformer une image en une forme numérique et à effectuer certaines opérations pour en tirer des informations utiles. Le système de traitement d'images traite généralement toutes les images comme des signaux 2D en appliquant certaines méthodes prédéterminées de traitement des signaux **[w07].**

7-1-Définition d'une image :

Une image est une représentation visuelle de quelque chose : comme une ressemblance d'un objet produite sur un support photographique ou une image produite sur un affichage électronique (tel qu'un écran de télévision ou d'ordinateur)

Une image est stockée en mémoire sous forme de points de base appelés pixels. On peut penser à une image numérique comme une page de nombres organisée dans un tableau ou une matrice de telle sorte que chaque nombre représente les caractéristiques du pixel **[15]**.

On peut représenter la position de chaque pixel par les coordonnées sur l'axe X horizontal et l'axe Y vertical figurant dans l'image suivante.

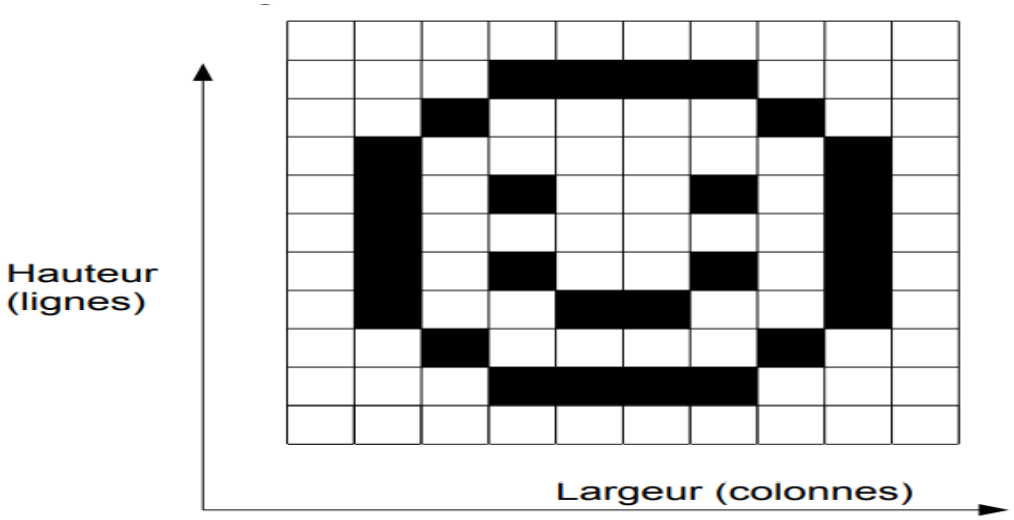


Figure 1-10 : une image sous forme de pixels **[f05]**

Par exemple, si les dimensions d'une image sont de 500 x 400 (largeur x hauteur), le nombre total de pixels de l'image est de 200000.

Ce pixel est un point de l'image qui prend une teinte, une opacité ou une couleur spécifique. Il est généralement représenté de l'une des manières suivantes **[w07]**:

7-1-1-Les types d'image :

7-1-1-1-Image binaire :

Les images qui n'ont que deux valeurs uniques d'intensité de pixel - 0 (représentant le noir) et 1 (représentant le blanc) - sont appelées images binaires. Ces images sont généralement utilisées pour mettre en évidence une partie discriminante d'une image colorée. Par exemple, elles sont couramment utilisées pour la segmentation d'images, comme illustré ci-dessous. **[w07]**

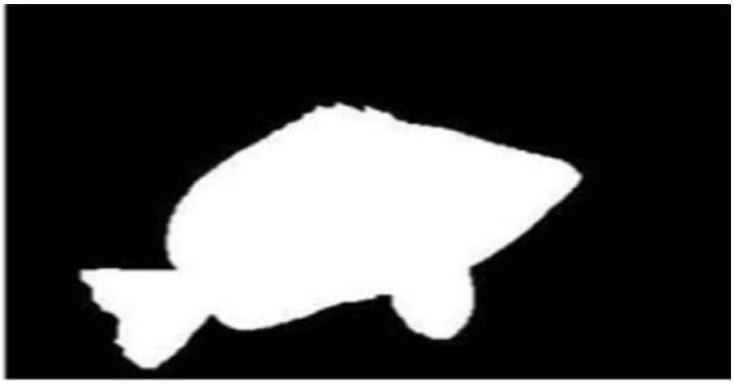


Figure 1-11-une image en binaire **[f23]**

7-1-1-2-Les images en couleur (Red, Green, Blue):

Une image en couleur est contituée de trois couches : une couche rouge (R), une couche verte (V), une couche bleue (B). Nous n'allons pas ici expliquer comment sont représentées les couleurs. Un pixel est composé de trois nombres entiers compris entre 0 et 255 (les nombres entiers représentent l'intensité du rouge, du vert et du bleu).

Pour l'image en couleur complète, il y a donc 24 bits par pixels, à multiplier par le nombre de pixels pour obtenir l'occupation totale en mémoire. **[w17]**

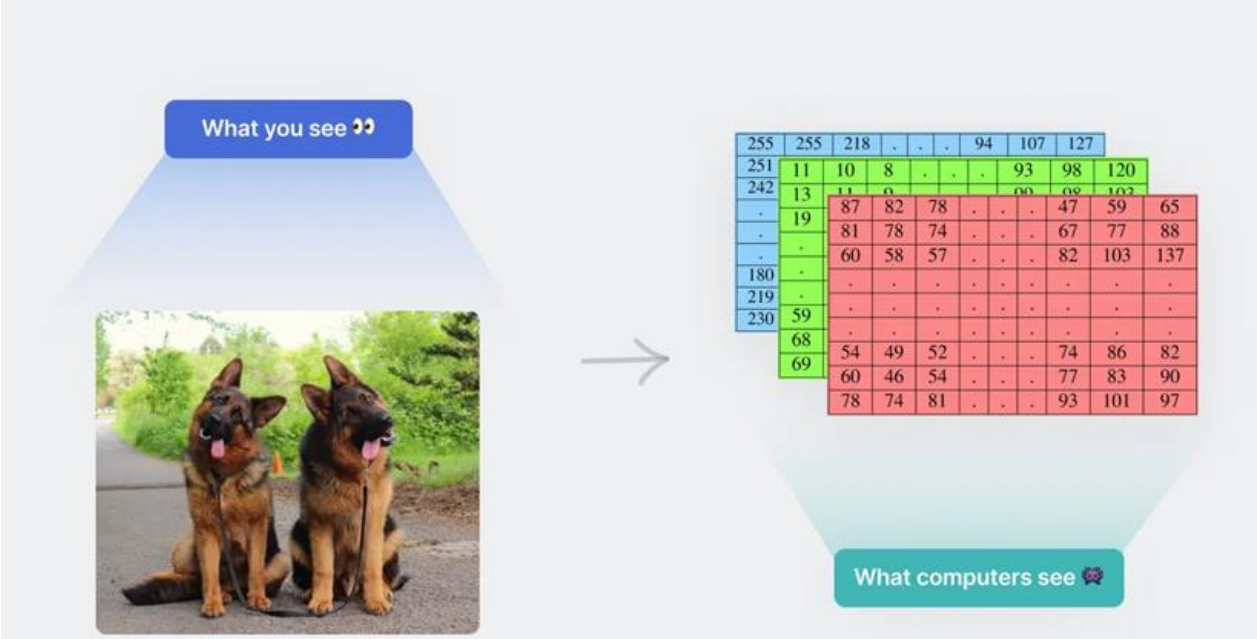


Figure 1-12-exemple d'une image en couleur RVB **[f21]**

7-1-1-3-Les images ARGB (Alpha, Red, Green, Blue) :

ARGB est un modèle de couleur qui inclut un canal supplémentaire pour la transparence alpha en plus des canaux rouge, vert et bleu. Le canal alpha détermine la transparence d'un pixel, 0 représentant une transparence totale et 255 une opacité totale. Cela permet de créer des couleurs semi-transparentes ou translucides dans les graphiques numériques.

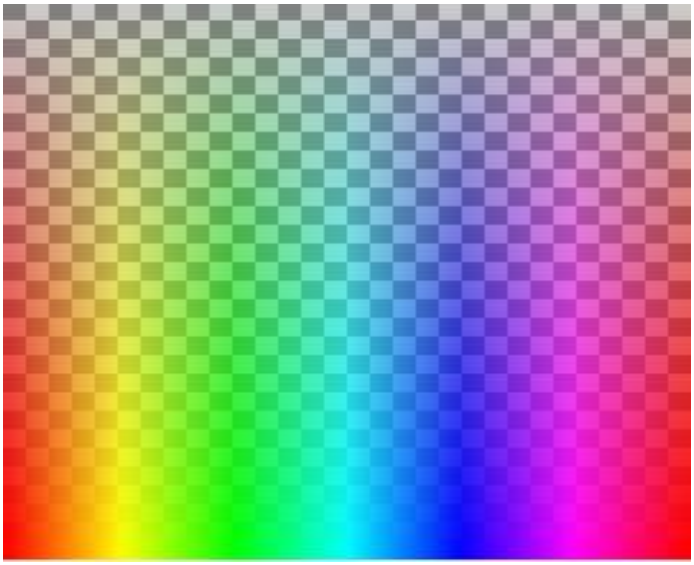


Figure 1-13- Exemple d'une image RGBA composée sur un fond en damier. L'alpha est de 0 % en haut et de 100 % en bas. **[f22]**

7-1-1-4-Les images en niveau de gris :

Une image en niveaux de gris est une image dans laquelle les seules couleurs sont des nuances de gris. La raison pour laquelle ces images se distinguent de tout autre type d'image en couleur est qu'il y a moins d'informations à fournir pour chaque pixel.

Dans une image à niveaux de gris, chaque pixel a une valeur comprise entre 0 et 255, où zéro correspond au « noir » et 255 au « blanc ». Les valeurs comprises entre 0 et 255 sont des nuances de gris, les valeurs proches de 0 étant plus foncées et les valeurs proches de 255 étant plus claires. **[w18]**.

Figure 1-14-image RGB à gauche et en niveau de gris à droite **[f22]**

8-Quelques techniques de traitement d'images :

Le traitement d'image peut être utilisé pour améliorer la qualité d'une image, supprimer des objets indésirables d'une image ou même créer de nouvelles images à partir de zéro. Par exemple, le traitement d'image peut être utilisé pour supprimer l'arrière-plan d'une image d'une personne, ne laissant que le sujet au premier plan.

Dans cette section, nous nous concentrerons sur certaines des tâches de traitement d'images les plus courantes**[w18]**.

8-1-Amélioration de l'image (Image Enhancement) :

L'une des tâches de traitement d'images les plus courantes est l'amélioration de la qualité d'une image. Elle a des applications cruciales dans les tâches de vision par ordinateur, de télédétection et de surveillance. Une approche courante consiste à ajuster le contraste et la luminosité de l'image. **[w18]**.

Figure 1-15- Exemple d'opération d'amélioration de l'image **[f22]**

8-2-Restauration d'images :

La qualité des images peut se dégrader pour plusieurs raisons, en particulier les photos datant de l'époque où le stockage en nuage n'était pas si courant. Par exemple, les images numérisées à partir de copies papier prises avec de vieux appareils photo instantanés sont souvent rayées.

De puissants algorithmes de restauration d'images basés sur l'apprentissage profond pourraient être en mesure de révéler de gros morceaux d'informations manquantes dans des documents déchirés. **[w18]**.

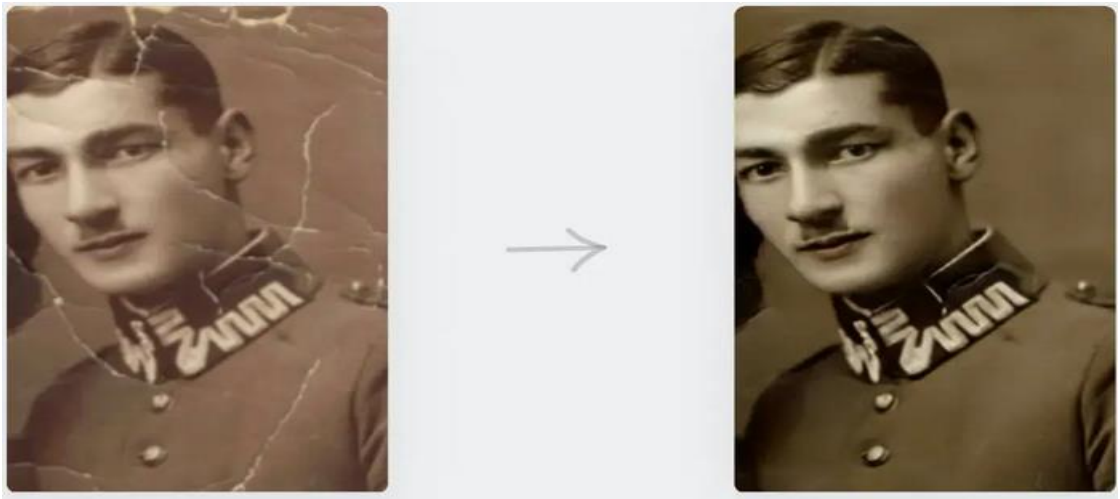


Figure 1-16- Exemple d'opération de restauration d'image**[f22]**

8-3-**Compression d'images :**

La compression d'images est le processus qui consiste à réduire la taille du fichier d'une image tout en essayant de préserver la qualité de l'image. Cela permet d'économiser de l'espace de stockage, notamment pour exécuter des algorithmes de traitement d'images sur des appareils mobiles et périphériques, ou de réduire la bande passante nécessaire à la transmission de l'image

Les approches traditionnelles utilisent des algorithmes de compression avec perte, qui réduisent légèrement la qualité de l'image afin d'obtenir un fichier plus petit. **[w18]**

Figure 1-17- Exemple d'opération de compression d'image**[f22]**

8-4-Manipulation d'images :

La manipulation d'images consiste à modifier une image pour en changer l'apparence. Cette opération peut être souhaitée pour plusieurs raisons, comme la suppression d'un objet indésirable d'une image ou l'ajout d'un objet qui n'est pas présent dans l'image. Les graphistes procèdent souvent ainsi pour créer des affiches, des films, etc.

Un exemple de manipulation d'image est le transfert de style neuronal, une technique qui utilise des modèles d'apprentissage profond pour adapter une image au style d'une autre. Par exemple, une image normale peut être transférée dans le style de la « Nuit étoilée » de van Gogh. Le transfert de style neuronal permet également à l'IA de générer de l'art. **[w18]**.

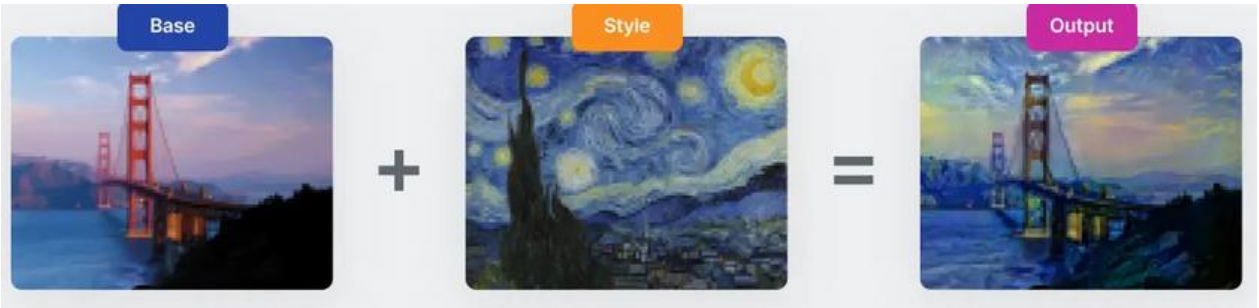


Figure 1-18- Exemple d'opération de manipulation d'image**[f22]**

8-5-Classification d'images :

La classification d'images fait référence à la tâche consistant à attribuer une étiquette ou un tag à une image. Généralement, des algorithmes d'apprentissage profond supervisés sont utilisés pour les tâches de classification d'images et sont entraînés sur des images annotées avec une étiquette choisie parmi un ensemble fixe d'étiquettes prédéfinies. Les annotations requises pour la classification d'images se présentent sous la forme de

simples étiquettes textuelles, de numéros de classe ou d'encodages à une touche où une liste zéro contenant tous les identifiants uniques possibles est formée - et un élément particulier de la liste basé sur l'étiquette de la classe est mis à un. Souvent, d'autres formes d'annotations sont converties sous forme d'encodage unique ou d'ID de classe avant que les étiquettes ne soient utilisées dans les fonctions de perte correspondantes**.[w01]**

Figure 1-19- Exemple d'opération de classification d'mage **[f01]**.

8-6-Détection et reconnaissance d'objets :

La détection d'objets (parfois appelée reconnaissance d'objets) est la tâche qui consiste à détecter des objets dans une image. Les annotations pour ces tâches se présentent sous la forme de boîtes englobantes et de noms de classe, les coordonnées extrêmes des boîtes englobantes et l'identifiant de la classe étant considérés comme la vérité de base.
Cette détection se présente sous la forme de boîtes englobantes où le réseau détecte les coordonnées des boîtes englobantes de chaque objet et les étiquettes de classe correspondantes. [w01]

Figure1-20- Exemple d'opération de détection d'objet **[f01]**.

8-7-Segmentation d'image:

La segmentation d'images est la tâche qui consiste à segmenter les régions d'une image en fonction de leur appartenance à une classe ou à un label particulier.

Il existe plusieurs types de segmentation d'images parmi lesquelles la segmentation de contour qui nous permet de reconnaître les limites ou les lignes des objets (ce que l'on appelle la "reconnaissance des limites") dans une image. Ces limites peuvent inclure les bords d'un objet individuel ou la zone de topographie affichée dans l'image. Ce type d'annotation est utilisé dans l'apprentissage des voitures autonomes. Il est utilisé pour reconnaître les limites des trottoirs, par exemple, ou des voies de circulation.**[w01]**

Figure1-21-Exemple d'opération de segmentation d'image**[f08].**

9-L'annotation d'images et la langue des signes :

L'annotation d'image, l'image captionning , le sous-titrage automatique d'image ou étiquetage sont définis comme le processus de génération de légendes ou de descriptions textuelles pour les images sur la base du contenu de l'image. Il s'agit d'une tâche d'apprentissage automatique qui implique à la fois le traitement du langage naturel (pour la génération du texte) et la vision par ordinateur (pour la compréhension du contenu de l'image).

Le jeu de données sera sous la forme [image → caption]. **[01]**

Dans ce travail on va se focaliser sur l'annotation des signes de langue américaine (American Sign Language – ASL) par le deep learning .

Les images peuvent être annotées avec des gestes de la langue des signes pour aider les personnes muettes à comprendre le contenu visuel. Les annotations peuvent représenter les gestes de la langue des signes correspondant à des mots ou des phrases spécifiques présents dans l'image.

9-1-La Langue des Signes Américaine (ASL) :

La langue des signes américaine (ASL) est une langue qui possède ses propres règles et sa propre grammaire, exprimées par des mouvements de la main. Elle sert de langue principale aux personnes sourdes ou malentendantes. Il existe diverses langues des signes dans le monde, avec des variantes telles que la langue des signes américaine (ASL), qui diffère de la langue des signes britannique (BSL). L'ASL comprend 26 lettres de l'alphabet anglais, représentées de manière unique par des formes de mains. L'apprentissage automatique, en particulier les réseaux neuronaux, joue un rôle essentiel dans la détection de l'alphabet ASL, en aidant à prédire les lettres sur la base de données visuelles.

Les lettres alphabétiques sont presque la première chose qu'un débutant apprend en langue des signes, et ce pour une bonne raison. Lorsqu'un apprenant ne connaît pas le signe ASL, il peut épeler le mot anglais. Bien entendu, cette méthode fonctionne lorsque le sourd est bilingue dans les deux langues, c'est-à-dire l'ASL et l'anglais. **[w2]**

figure1-22-Alphabet du Langage de Signes Américain **[f01]**

9-2-Les domaines d'application d'annotation des signes :

L'annotation des signes de langue américaine (American Sign Language - ASL) par le deep learning trouve des applications dans divers domaines. Voici quelques exemples :

9-2-1-Traduction en temps réel :

Les systèmes d'annotation des signes de l'ASL basés sur le deep learning peuvent être utilisés pour traduire en temps réel les signes effectués en ASL vers la langue parlée ou écrite. Cela facilite la communication entre les personnes sourdes qui utilisent l'ASL et les personnes entendantes dans des situations de conversation en face à face ou en ligne.

9-2-2-Assistance à l'apprentissage de l'ASL :

Les systèmes d'annotation des signes de l'ASL peuvent être utilisés comme outils d'apprentissage interactifs pour les personnes qui souhaitent apprendre l'ASL. Ces systèmes peuvent fournir des commentaires en temps réel sur la précision des gestes effectués par l'apprenant, ce qui facilite l'apprentissage et la pratique de l'ASL.

9-2-3-Accessibilité numérique :

L'annotation des signes de l'ASL peut être utilisée pour rendre les contenus numériques accessibles aux personnes sourdes qui utilisent l'ASL. Par exemple, les vidéos en ligne peuvent être automatiquement sous-titrées en langue des signes américaine, permettant ainsi aux personnes sourdes de comprendre le contenu vidéo sans avoir besoin de traductions écrites.

9-2-4-Communication en ligne :

Les systèmes d'annotation des signes de l'ASL peuvent être intégrés aux plateformes de communication en ligne, telles que les applications de messagerie et les réseaux sociaux, pour permettre aux utilisateurs de communiquer en utilisant l'ASL. Cela facilite la communication entre les personnes sourdes utilisant l'ASL et les personnes entendantes dans un environnement en ligne.

9-2-5-Reconnaissance de gestes pour les interfaces homme-machine :

Les techniques d'annotation des signes de l'ASL peuvent être utilisées pour développer des interfaces homme-machine basées sur la reconnaissance de gestes. Par exemple, les gestes de la langue des signes américaine peuvent être utilisés pour contrôler des dispositifs électroniques, tels que des ordinateurs, des smartphones ou des robots, offrant ainsi une nouvelle façon d'interagir avec ces technologies.

10-Problématique étudié et travaux connexes :

Nous allons nous intéresser à la problématique de la description d'image qui est l'un des problèmes fondamentaux de la vision par ordinateur qui a une grande variété d'applications pratiques.

La description d'images et l'annotation de la langue des signes américaine (American Sign Language - ASL) sont des sujets passionnants et importants pour l'accessibilité et la communication avec les personnes sourdes ou malentendantes.

Pour l'annotation de la langue des signes américaine, l'apprentissage profond peut être utilisé pour reconnaître et interpréter les gestes de la langue des signes**.**

Pour aborder cette problématique, il existe plusieurs travaux connexes à la conversion de la langue des signes Américaine en texte. Voici quelques-uns

Référence	**Méthodologie**	**Description**
Deep Learning Technology to Recognize American Sign Language Alphabet by Bader Alsharif, Ali Salem Altaher Ahmed Altaher Mohammad Ilyas and Easa Alalwany**[08]**	Méthodologie basée sur ResNet-50,EfficientNet ConvNeXt et AlexNet.	Les résultats expérimentaux de cette étude ont révélé que ResNet-50 a atteint un taux de précision exceptionnel de 99,98 %, le plus élevé de tous les modèles. EfficientNet a atteint un taux de précision de 99,95 %, ConvNeXt a atteint une précision de 99,51 %, AlexNet a atteint une précision de 99,50 %,
Deep Learning Technology to Recognize American Sign Language Alphabet by Bader Alsharif, Ali Salem Altaher Ahmed Altaher Mohammad Ilyas and Easa Alalwany**[08]**	L'architecture ViT est composée de deux éléments principaux : une couche d'incorporation de patch et une couche d'encodage basée sur un transformateur.	leur modèle personnel VisionTransformer a obtenu la précision la plus faible, à savoir 88,59 %.
Al-Obodi, A.H.; Al-Hanine, A.M.; Al-Harbi, K.N.; Al-Dawas, M.S.; Al-Shargabi, A.A. A Saudi Sign Language	Une architecture basée sur un CNN	les auteurs ont développé un système de reconnaissance de la langue des signes arabe (ArSL) basé sur un CNN

recognition system based on convolutional neural networks. **[09]**		Ils ont utilisé un ensemble de données composé de 40 signes arabes, chaque signe comportant 700 images différentes Le résultat a montré une précision de 97,69% pour les données de formation et de 99,47% pour les données de test.
Hayani, S.; Benaddy, M.; El Meslouhi, O.; Kardouchi, M. Arab sign language recognition with convolutional neural networks**[10]**	Basée sur une architecture légère du CNN.	Ont présenté un système de reconnaissance ArSL hors ligne basé sur un modèle de réseaux neuronaux convolutionnels profonds qui peut reconnaître automatiquement les lettres et les chiffres de un à dix. Ils ont utilisé un ensemble de données réelles composé de 7869 images RVB. Le système proposé a atteint une précision de 90,02 %**.**
Kasapbaşi, A.; ELBUSHRA, A.E.A.; Omar, A.H.; Yilmaz, A. DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals. *Comput. Methods Programs Biomed. Update* 2022, *2*, 100048**[11]**	Un simple système CNN de reconnaissance du langage des signes	les auteurs ont créé un ensemble de données et un système CNN de reconnaissance du langage des signes pour interpréter l'alphabet gestuel des signes américains et le traduire dans notre langue naturelle. La précision globale pour l'ensemble de données basée sur le modèle CNN est de 99 %.
Mohandes, M.; Liu, J.; Deriche, M. A survey of image-based arabic sign language recognition. In Proceedings of the 2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14), Barcelona, Spain, 11–14 February 2014. **[12]**	Une implémentation du CNN.	Pour un autre système de reconnaissance de la langue des signes proposé, les auteurs ont entraîné un modèle d'apprentissage profond CNN à reconnaître 87 000 images ASL . Ils ont réussi à obtenir une accuracy de 78,50 %.
Cayamcela, M.E.M.; Lim, W. Fine-tuning a pre-trained convolutional neural network model to translate American sign language in real-time. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; IEEE: Piscataway, NJ, USA, 2019;	Deux architectures de réseaux neuronaux différentes, AlexNet et GooLeNet.	Dans le même contexte, [38] a utilisé le même ensemble de données de 87 000 images pour la classification. Ils ont utilisé les modèles (AlexNet et GooLeNet), et leurs résultats globaux de formation étaient de 99,39 % pour AlexNet et de 95,52 % pour GoogLeNet**.**

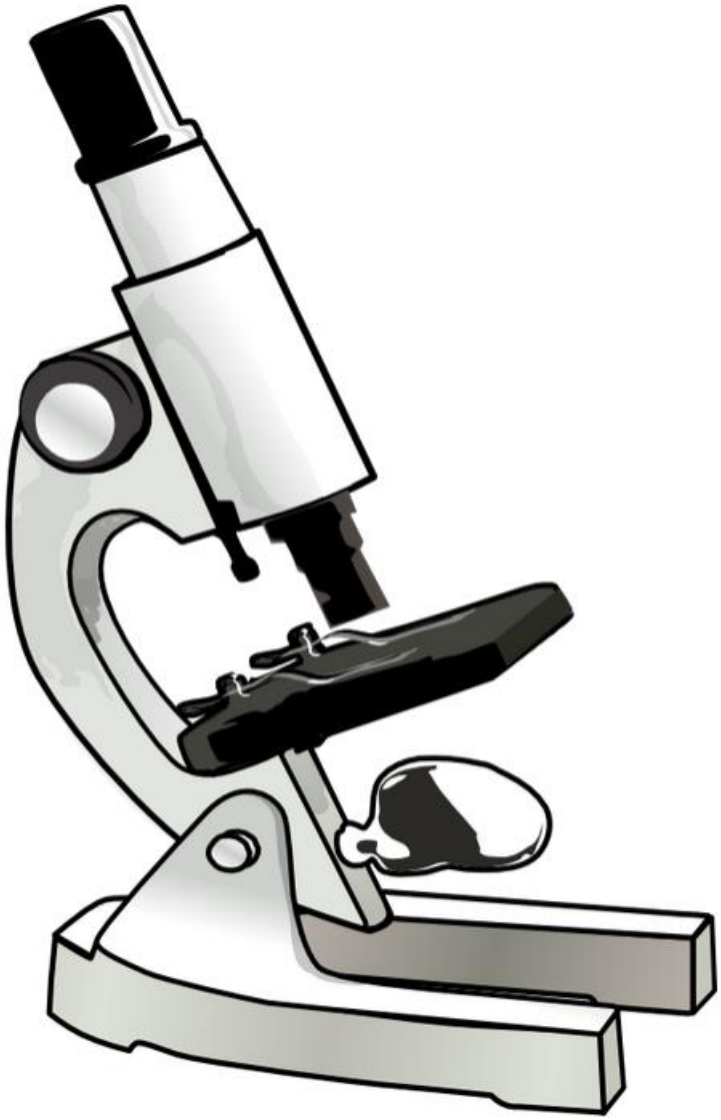
pp. 100–104. [**Google Scholar**] **[13]**		
Ma, Y.; Xu, T.; Kim, K. Two-Stream Mixed Convolutional Neural Network for American Sign Language Recognition. *Sensors* 2022, *22*, 5959. [Google Scholar] **.[14]**	Deux architectures de réseaux neuronaux différentes, AlexNet et ResNet-50	Dans [39], les auteurs ont évalué la reconnaissance de l'alphabet ASL en utilisant deux architectures de réseaux neuronaux différentes, AlexNet et ResNet-50, en utilisant le même ensemble de données que nous avons utilisé. Les résultats ont montré qu'AlexNet atteignait une précision de 94,74 %, tandis que ResNet-50 le surpassait de manière significative avec une précision de 98,88 %.**[14]**
11] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham. **[15]**	Méthodologie basée sur un réseau convolutif simple (CNN).	Pigou et al., [11] ont présenté une méthode neuronale convolutive pour identifier les signes de langue. Ils ont utilisé un dataset linguistiques (ASL) composé de plus de 1 400 vidéos, chacune présentant un seul signe. Plusieurs couches de convolution et de mise en commun ont été utilisé dans leur modèle CNN suggéré, qui a été suivi de couches entièrement liées pour la classification. Sur leur ensemble de tests, ils étaient précis à 87,4 %, surpassant les procédures de pointe antérieures à l'époque.Les couches convolutives et l'augmentation étaient les aspects les plus significatifs impactant les performances du modèle des auteurs, selon une étude de sensibilité qu'ils ont réalisée pour analyser l'importance des divers composants du modèle**.**

Tableau 1.4 : Les travaux connexes sur l'annotation du langage du signe ASL.

12-Conclusion :

En conclusion de ce chapitre, nous avons examiné la notion d'image, ses différent types, l'annotation d'image, le traitement d'image, un aperçu global sur l'intelligence artificielle et sa taxonomie ainsi que les travaux connexes qui vont constituer une plateforme pour nos modèles d'annotation visés.

Dans le prochain chapitre, nous approfondirons davantage les applications spécifiques de l'architecture CNN dans le contexte de notre travail.



Chapitre02 :

Les réseaux de neurones convolutifs

1-Introduction :

Dans cette partie, nous allons nous focaliser sur un des algorithmes les plus performants du Deep Learning: les Convolutional Neural Network (CNN) ou réseaux de neurones convolutifs en français. Ce sont des modèles de programmation puissants permettant notamment la reconnaissance d'images en attribuant automatiquement à chaque image fournie en entrée, une étiquette correspondant à sa classe d'appartenance.

2-Pourquoi le CNN ?

Les **CNN** désignent une sous-catégorie de réseaux de neurones et sont à ce jour un des modèles de classification d'images réputés être les plus performant.

Leurs applications vont de la reconnaissance d'images et de vidéos à la classification d'images, en passant par l'analyse d'images médicales, la vision par ordinateur et le traitement du langage naturel.

3-Lexique essentiel pour comprendre le (CNN) :

Avant de se lancer dans la compréhension en profondeur des réseaux de neurones convolutifs (CNN), il est essentiel de s'approprier le lexique de base de cette architecture neuronale. Comme tout domaine spécialisé, les CNN ont leur propre terminologie qui peut paraître intimidante de prime abord. Pourtant, une fois familiarisé avec ces concepts clés, la compréhension globale du fonctionnement et de la structure des CNN devient beaucoup plus accessible.

3-1-Les Couches du CNN :

L'architecture d'un réseau neuronal artificiel est définie par une collection d'unités connectées ou de nœuds (neurones).l'ANN est composé de trois couches principales : l'entrée, la couche cachée et la sortie, mais dans le cas de l'apprentissage profond, le nombre de couches peut atteindre des milliers, et certaines ANN d'apprentissage profond ont un nombre de couches inconnu , c'est pourquoi on l'appelle "Deep" (profond). **[17]**

3-1-1- Couche d'entrée :

Il y a une couche d'entrée dans un ANN, chaque type d'ANN reçoit différents types de données (informations) en fonction de l'apprentissage visé, par exemple, dans le domaine de la vision par ordinateur, la première couche est constituée de neurones contenant les intensités des pixels. **[17]**

3-1-2-Couches cachées:

Chaque réseau est unique, il n'y a pas de nombre spécifique de couches, certains réseaux utilisent un nombre inconnu de couches entre l'entrée et la sortie, d'où leur nom de couches cachées.

Le nombre de couches cachées dépend du type de données, de leur taille et de l'apprentissage visé, par exemple, pour la détection des visages dans le CNN, le réseau a besoin d'une couche pour détecter les arêtes (diagonales, verticales ...), une seconde pour détecter la combinaison des arêtes, qui sont ensuite combinées pour former un réseau., une troisième pour détecter les caractéristiques du visage (yeux, nez, ...) et une autre pour détecter la combinaison des caractéristiques et décider s'il s'agit ou non d'un visage. **[17]**

3-1-3-Couche de sortie :

La couche de sortie contient les résultats de la classification, par exemple pour la vision par ordinateur et la reconnaissance des chiffres de l'écriture manuscrite, la couche de sortie contient 10 neurones en fonction du nombre de chiffres utilisés (0,1,2,...,9).

le nombre de chiffres utilisés (0,1,2,...9), après les nombreux calculs effectués dans les couches cachées, la dernière couche fournit une estimation de ce que pourrait être le chiffre dans l'entrée.

La décision repose sur fonctions mathématiques appelées fonctions d'activation (tableau 2.1).

3-2-Les poids du CNN :

Les réseaux neuronaux relient les neurones d'une couche à l'autre par des associations pondérées.

Les réseaux apprennent (ou sont entrainés) en traitant des exemples contenant des paires d'"entrées" et de "sorties" connues, en formant des associations pondérées par des probabilités entre les deux, qui sont stockées dans la structure de données du réseau lui-même.

Le réseau ajuste ensuite ses associations pondérées selon une règle d'apprentissage et en utilisant la valeur de l'erreur. Des ajustements successifs amèneront le réseau neuronal à produire une sortie, c'est-à-dire une valeur d'erreur.

Le réseau neuronal produit alors une sortie de plus en plus proche de la sortie cible. Après un nombre suffisant de ces ajustements, la formation peut être interrompue en fonction de certains critères. Le résultat est appelé modèle **[17].**

3-3-Les Fonctions d'activation :

Un nœud (neurone) donné prend la somme pondérée de ses entrées et la fait passer par une fonction d'activation non linéaire. C'est la sortie du nœud, qui devient alors l'entrée d'un autre nœud dans la couche suivante.

La fonction d'activation est ce qui décide si le neurone est activé ou non, par exemple, dans le domaine de la vision par ordinateur et du traitement d'images si deux neurones détectent deux arêtes différentes qui forment un coin, et que l'un des neurones de la couche suivante est spécifié pour ce coin, la fonction d'activation est activée.

pour ce coin, la fonction d'activation "activera" le neurone après avoir reçu les deux

signaux positifs et enverra évidemment un signal à la couche suivante en mentionnant qu'il y a un coin à cet endroit précis **[17]**.

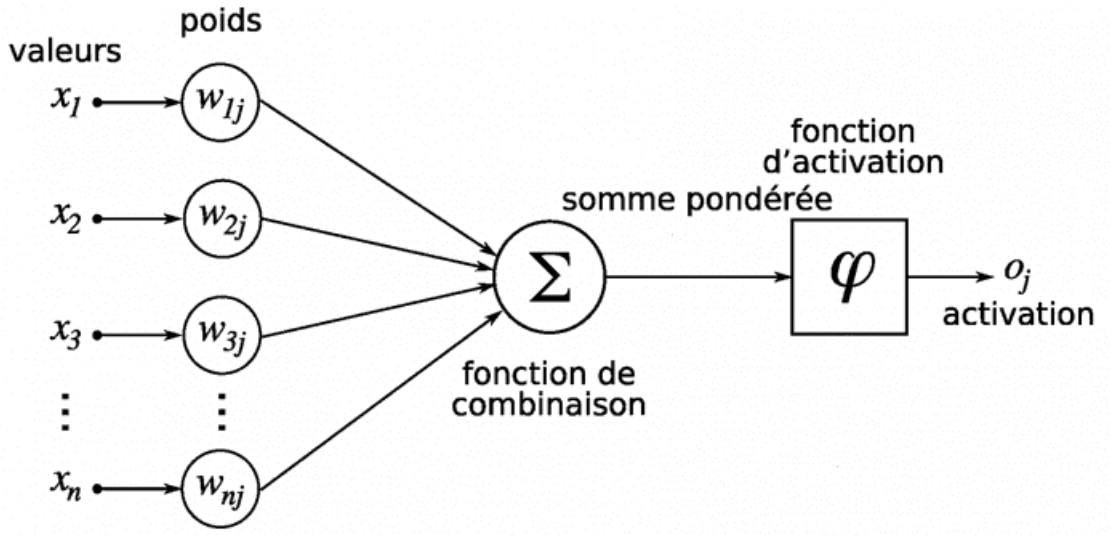


Figure2-1-fonctionnement d'une fonction d'activation **[f14]**

-De nombreuses fonctions d'activation ont été utilisées, chacune ayant sa propre formule d'où des résultats différents.

Bien que les fonctions puissent toutes être utilisées dans le même but, le résultat de l'apprentissage varie,Pour choisir la bonne fonction, il faut donc effectuer des tests et les comparer.[18]

Nom	**Fonction et graphique**	**Avantages et inconvénients**
2-Step(Binaire)	$f(x) = \{$ 0 Si $x < 0$ 1 Si $x \geq 0$	-Classification binaire -Ne fonctionne pas en classification multi-label - La dérivée pour le calcul du gradient est toujours 0, donc impossible de mettre à jour les poids
Linear	$(x) = a$x	-Classification binaire et multiclasse - Hautement interprétable - La dérivée correspond à "a", de sorte que la mise à jour des poids et des biaises pendant la rétro-progation sera constante. biaises pendant la rétro-progation sera constante. - Pas efficace si le gradient est toujours le même.
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$ $= \frac{e^x}{e^x+1}$	-La sortie de chaque neurone peut saturer. -La meilleure sensibilité se situe autour du point central (0, 0.5). -L'algorithme ne peut pas apprendre pendant la saturation (c'est la source source du problème du gradient qui s'évanouit, correspondant à l'absence de

		direction dans le gradient).
Tanh	$(x) = 2 * S(2x) - 1$	-La plage [-1,1] -Le gradient est plus fort que celui de la sigmoïde (les dérivées sont plus raides). - Comme la sigmoïde, le tanh a également un problème de gradient de disparition.
ReLU	$(x) = \max(0, x)$	□ Facile à mettre en œuvre et très rapide □ Véritable valeur 0 □ L'optimisation est facile lorsque la fonction d'activation est linéaire □ Le plus utilisé dans l'écosystème des réseaux neuronaux -Elle ne peut pas être différentiable lorsque x = 0. La descente du gradient ne peut pas être calculée pour ce point. être calculée pour ce point - "problème de ReLU mourant" -Ne convient pas aux algorithmes de classe RNN (RNN, LSTM, GRU)
Leaky ReLU	$f(x) = \begin{cases} 0.01x & if\ x < 0 \\ x & if\ x \geq 0 \end{cases}$	-Corriger le "problème de la ReLU mourante" - Même compartiment de la fonction d'activation ReLU pour la partie y=x

Parametric ReLU	$f(x) = \begin{cases} ax \; if \; x < 0 \\ x \quad if \; x \geq 0 \end{cases}$	-Généraliser la fonction d'activation ReLU -Éviter le "problème de la ReLU mourante". - Le paramètre "a" est appris par le réseau neuronal
e-ReLU	$f(x) = \begin{cases} a(e^x - 1) \; if \; x < 0 \\ x \qquad\quad if \; x \geq 0 \end{cases}$	-Se lisse lentement jusqu'à ce que sa production soit égale à -α alors que RELU lisse brusquement. - Alternative solide à la ReLU. - Contrairement à ReLU, ELU peut produire des sorties négatives. - Pour x > 0, il peut faire exploser l'activation avec une plage de sortie de [0, ∞].
Softmax	$f(x) = \frac{e^x}{\sum_{j=1}^{K} e^{x_j}}$	-Il s'agit d'une distribution de probabilité, la sortie est donc différente si l'on tient compte de la somme de l'exponentielle. en tenant compte de la somme des exponentielles.

Tableau 2.1 : Les fonctions d'activation avec leurs avantages et leurs inconvénients.

3-4-La rétro-propagation(back propagation):

Dans un réseau de neurone lorsque le résultat réel est différent du résultat attendu, les poids appliqués aux neurones sont mis à jour. Parfois, les résultats attendus et réels se situent dans les limites du seuil d'erreur et le réseau de neurones est considéré comme optimal.

Cependant, la sortie attendue est parfois différente de la sortie réelle. En conséquence, les informations sont renvoyées dans le réseau et les poids et biais sont améliorés. Ce processus est de nature récursive et est appelé rétro-propagation.

Le processus de rétro-propagation permet aux algorithmes d'auto-apprendre ainsi, la rétro-propagation rend les réseaux de neurones intelligents en s'améliorant automatiquement. **[w13]**

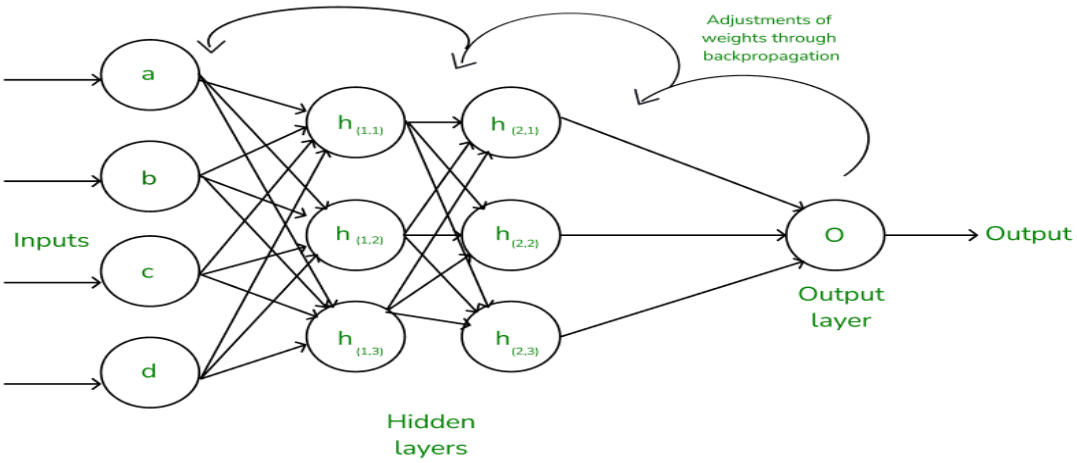


figure2-2- Illustration simple du fonctionnement de la rétropropagation par ajustement des poids**[f14]**

4-Les blocs de construction d'un CNN :

Il existe quatre types de couches pour un réseau neuronal convolutif : la couche convolutive, la couche de mise en commun, la couche de correction ReLU et la couche entièrement connectée.**[w14]**

Figure 2-3-L'architecture basique du CNN

4-1-La convolution :

La convolution est une opération mathématique simple généralement utilisée pour le traitement et la reconnaissance d'images. Son effet sur une image est similaire à celui du filtrage, qui fonctionne comme suit :

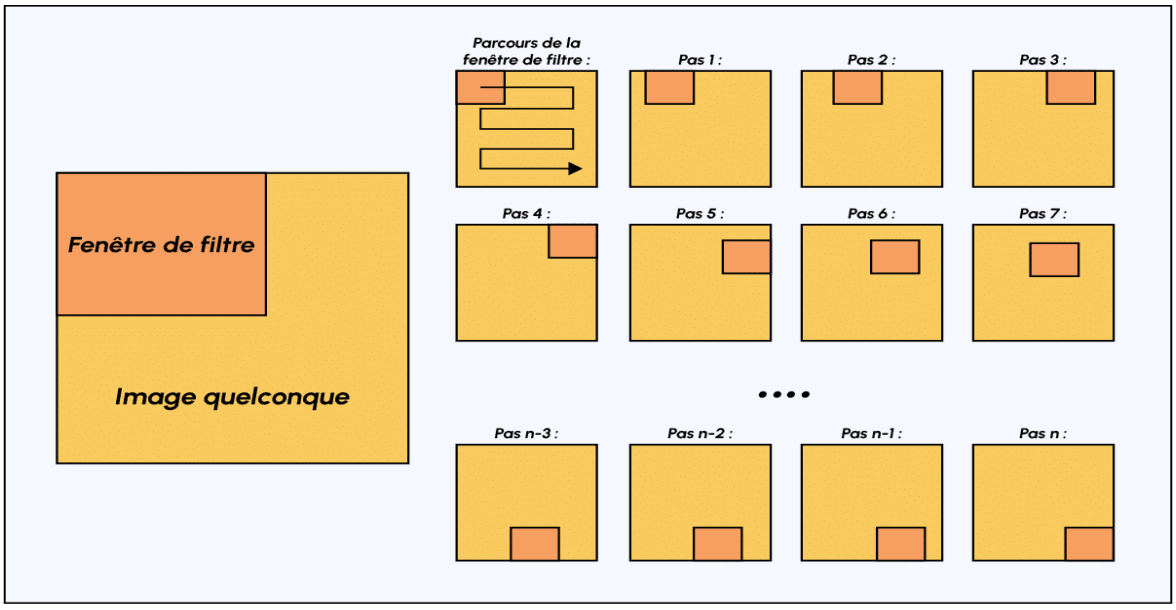


Figure2-4-Diagramme de déplacement de la fenêtre de filtrage sur l'image**[f09]**

Au départ, vous définissez la taille de la fenêtre de filtrage, en commençant généralement par le coin supérieur gauche, la fenêtre de filtrage, qui représente un élément, se déplace progressivement de gauche à droite d'un nombre prédéterminé de cellules (le pas) jusqu'à ce qu'elle atteigne la fin de l'image.

Pour chaque partie de l'image rencontrée, un calcul de convolution est effectué, ce qui permet d'obtenir une carte d'activation ou une carte de caractéristiques qui indique où se trouvent les caractéristiques dans l'image. Plus la valeur de la carte d'activation est élevée, plus la partie scannée de l'image ressemble à la caractéristique recherchée. Ce processus permet aux CNN d'identifier des modèles et des caractéristiques significatifs dans l'image d'entrée. **[w09]**

4-2-Le pooling :

Cette couche est chargée de réduire la dimensionnalité. Elle permet de réduire la puissance de calcul nécessaire au traitement des données. Le pooling peut être divisé en deux types : le max pooling et le pooling moyen. La valeur maximale de la zone couverte par le noyau sur l'image est renvoyée par le max pooling. La moyenne de toutes les valeurs de la partie de l'image couverte par le noyau est renvoyée par le pooling moyen.**[w12]**

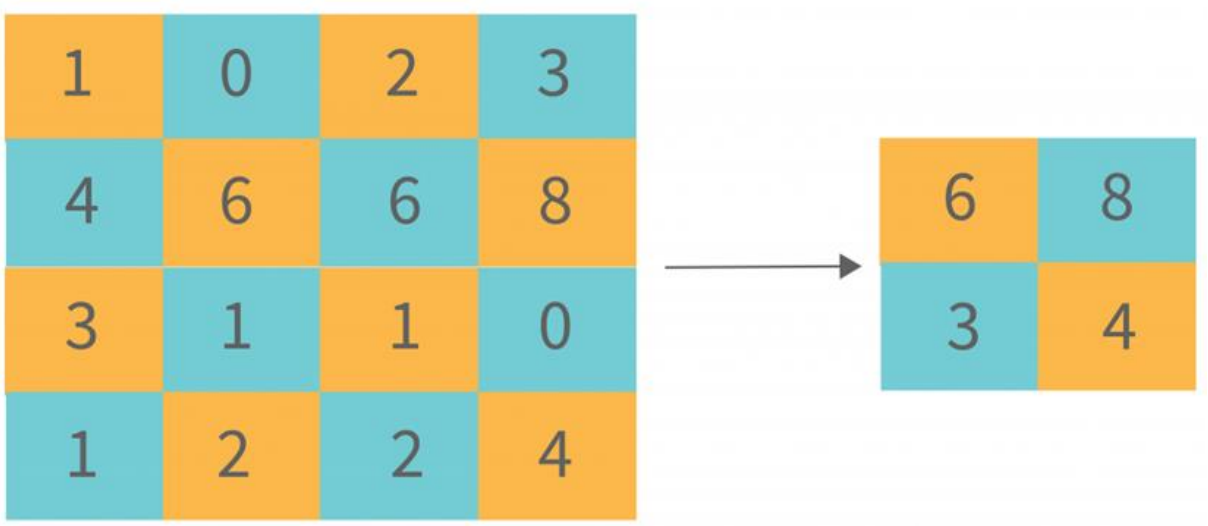


Figure2-5-Exemple d'une opération de pooling**[f12]**

4-3-Fully connected layer :

Une couche entièrement connectée, également appelée couche dense, est un type de couche utilisé dans les réseaux neuronaux artificiels où chaque neurone ou nœud de la couche précédente est connecté à chaque neurone de la couche actuelle. Elle est appelée

« entièrement connectée » en raison de ce lien complet. Les couches FC se trouvent généralement vers la fin de l'architecture d'un réseau neuronal et sont responsables de la production des prédictions de sortie finales. **[w12]**

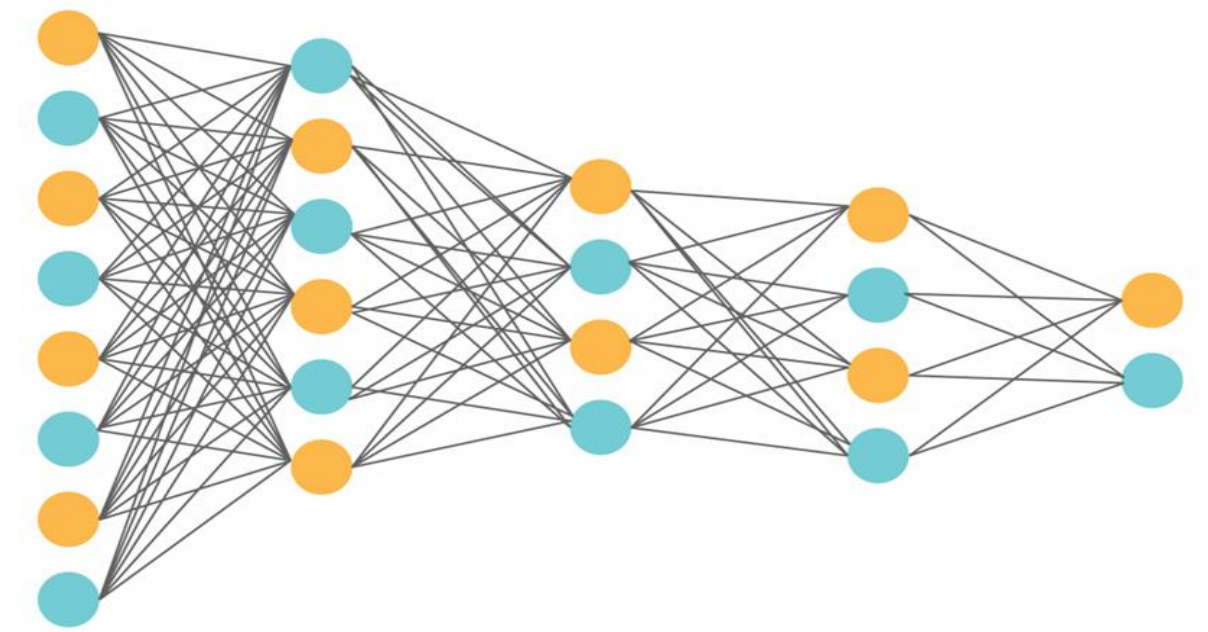


Figure 2-6- Couche entièrement connectée d'un réseau profond **[f13]**

4-4-La couche de correction ReLU (Rectified Linear Units):

ReLU (Rectified Linear Units) fait référence à la fonction non linéaire réelle définie par ReLU(x)=max(0,x). Visuellement, cela ressemble à ce qui suit :

La couche de correction ReLU remplace toutes les valeurs négatives reçues en entrée par des zéros. Elle agit comme une fonction d'activation.

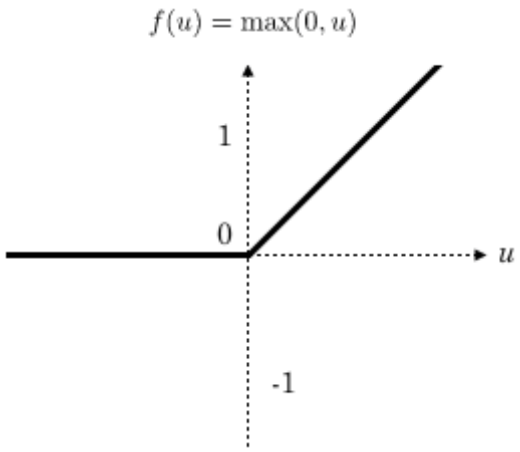


Figure 2-7- La fonction RELU **[f13]**

5-Etape pour l'implémentation d'un modèle basé sur le CNN:

Pour utiliser avec succès le CNN , il ne suffit pas de savoir comment le construire; il faut aussi connaître les étapes nécessaires pour les appliquer efficacement dans le monde réel. **[w10]**

Dans cet article, nous abordons le flux de travail d'un projet d'apprentissage profond, c'est-à-dire la manière dont nous élaborons des solutions d'apprentissage profond pour nous attaquer à des tâches réelles. Le flux de travail d'un modèle basé sur le CNN se compose de sept éléments principaux :

1. Acquisition de données
2. Prétraitement
3. Fractionnement et équilibrage de l'ensemble de données
4. Construction et entraînement du modèle
5. l'évaluation
6. Réglage des hyperparamètres
7. Déploiement de notre solution (pour l'industrie)

5-1 -Acquisition de données :

Dans un projet d'apprentissage profond, la préoccupation la plus pressante est presque toujours la taille du dataset, plus il est large plus notre modèle est performant. Notre capacité à acquérir des données peut faire de notre solution un succès ou un échec. Non seulement l'obtention de données est généralement la partie la plus importante d'un projet d'apprentissage profond, mais c'est aussi souvent la plus difficile. **[w10]**

5-2-Prétraitement;

Une fois notre dataset constitué, nous devons le prétraiter pour en faire des caractéristiques utiles à nos modèles d'apprentissage profond.

Le prétraitement des données dans l'apprentissage automatique fait référence à la technique de préparation (nettoyage et organisation) des données brutes pour les rendre adaptées à la construction et à l'entraînement de modèles d'apprentissage automatique. En d'autres termes, le prétraitement des données dans l'apprentissage automatique est une technique d'exploration des données qui transforme les données brutes en un format compréhensible et lisible. **[w11]**

5-3- Diviser et équilibrer le dataset :

Une fois que nous avons traité nos données, il est temps de diviser notre ensemble de données. En règle générale, nous divisons nos données en deux ensembles de données : entrainement et validation. Dans certains cas, nous créons également un troisième ensemble de données, appelé ensemble de test. Lorsque ce n'est pas le cas, nous utilisons souvent les termes "validation" et "test" de manière interchangeable.

Nous entraînons notre modèle sur l'ensemble de données d'entraînement et nous l'évaluons sur l'ensemble de données de validation. Si nous avons défini un troisième ensemble de test, nous testons notre modèle sur cet ensemble de données après avoir sélectionné notre modèle et réglé nos hyperparamètres.

Cette troisième étape nous permet d'éviter de choisir un ensemble d'hyperparamètres qui ne fonctionnent bien que sur les données que nous avons choisies pour notre ensemble de validation.

Nous conservons généralement 10 à 30 % de nos données à des fins de validation et de test. **[w10]**

5-4- Construction et entraînement du modèle :

Une fois que nous avons divisé notre ensemble de données, il est temps de choisir notre fonction de perte et nos couches.

Pour chaque couche, nous devons également sélectionner un nombre raisonnable d'unités cachées. Il n'y a pas de science absolue pour choisir la bonne taille de chaque couche, ni le nombre de couches - tout dépend de vos données et de votre architecture spécifiques.

Une bonne pratique consiste à commencer par quelques couches (2 à 6) et en général, nous créons chaque couche avec 32 à 512 neurones, nous avons également tendance à réduire la taille des couches cachées au fur et à mesure que nous progressons dans le modèle.

Nous commençons généralement par essayer les optimiseurs SGD et Adam et Lors de la définition d'un taux d'apprentissage initial, une pratique courante consiste à choisir par défaut 0,001. **[w10]**

5-5-Évaluation des performances:

Chaque fois que nous formons le modèle, nous évaluons ses performances sur notre ensemble de validation. Lorsque nous fournissons un ensemble de validation au moment de l'entraînement cela nous donnent une idée de la manière dont notre modèle se comportera sur de nouvelles données inédites.

Lorsque l'on considère les performances, il est important de choisir la bonne métrique. Si notre ensemble de données est fortement déséquilibré, la précision sera moins significative. Dans ce cas, il est préférable d'utiliser des mesures telles que la précision et le rappel. Le score F1 est une autre mesure utile qui combine à la fois la précision et le rappel. Une matrice de confusion peut aider à visualiser les points de données mal classés et ceux qui ne le sont pas. **[w10]**

5-6- Ajustement des hyperparamètres:

Nous aurons presque toujours besoin d'itérer sur nos hyperparamètres initiaux. Lors de l'entraînement et de l'évaluation de notre modèle, nous explorons différents taux d'apprentissage, tailles de lots, architectures et techniques de régularisation.

Au fur et à mesure que nous ajustons nos paramètres, nous devons surveiller nos pertes et nos mesures, et être à l'affût d'indices expliquant pourquoi notre modèle a du mal à s'adapter :

Un apprentissage instable signifie que nous devons probablement réduire notre taux d'apprentissage et/ou augmenter la taille de notre lot.

Une disparité entre les performances sur les ensembles d'apprentissage et d'évaluation signifie que nous sommes en train de surajuster (overfitting) et que nous devrions réduire la taille de notre modèle ou ajouter une régularisation (comme le dropout).

Des performances médiocres à la fois sur l'ensemble d'entraînement et sur l'ensemble de test signifient que nous sommes sous-adaptés et que nous avons besoin d'un modèle plus grand ou d'un taux d'apprentissage différent.

Une pratique courante consiste à commencer par un modèle plus petit et à augmenter nos hyperparamètres jusqu'à ce que les performances de formation et de validation divergent, ce qui signifie que nous nous sommes suradaptés à nos données.

Il est important de noter que les poids des réseaux neuronaux étant initialisés de manière aléatoire, vos résultats fluctueront, quels que soient les hyperparamètres. L'un des moyens d'obtenir des jugements précis consiste à exécuter plusieurs fois la même configuration d'hyperparamètres, avec différentes graines aléatoires.

Lorsque nos résultats sont satisfaisants, nous sommes prêts à utiliser notre modèle **[w10].**

5-7-Déploiement (pour l'industrie) :

Une fois que nous avons formé un modèle, nous pouvons vouloir le déployer dans le monde réel. C'est particulièrement vrai dans le secteur industriel, lorsque nos réseaux seront utilisés par nos collègues et nos clients **[w10].**

Note :

Ces lignes directrices ne sont pas des règles absolues. Au contraire, dans un projet d'apprentissage profond réussi, nous passons souvent d'une étape à l'autre, en ajustant et en déboguant continuellement nos scripts, nos données et nos architectures dans notre quête du modèle le plus performant.

6-L'Overfitting (Surapprentissage) du CNN:

L'overfitting se produit lorsque le modèle de CNN s'ajuste trop précisément aux données d'entraînement, perdant ainsi sa capacité de généralisation sur de nouvelles données.

Cela se traduit par de très bonnes performances sur l'ensemble d'entraînement, mais de mauvaises performances sur les ensembles de validation et de test**.[w03]**

6-1-Les symptômes de l'overfitting :

Voici les symptômes essentiels de l'overfitting du CNN :

- Écart important entre les performances d'entraînement et de validation/test.
- Courbe d'apprentissage qui continue à s'améliorer sur l'ensemble d'entraînement, mais stagne ou se dégrade sur l'ensemble de validation. **.[w03]**

6-2-Causes de l'overfitting dans les CNN:

- Modèle trop complexe par rapport à la taille de l'ensemble d'entraînement (trop de paramètres).
- Données d'entraînement insuffisantes ou trop homogènes.
- Absence ou mauvaise application de techniques de régularisation (L1/L2, dropout, augmentation de données, etc.).

7-L'Underfitting (Sous-apprentissage):

L'underfitting se produit lorsque le modèle de CNN n'a pas la capacité suffisante pour apprendre les motifs complexes présents dans les données.

Cela se traduit par de mauvaises performances, à la fois sur l'ensemble d'entraînement et sur les ensembles de validation/test**.[w03]**

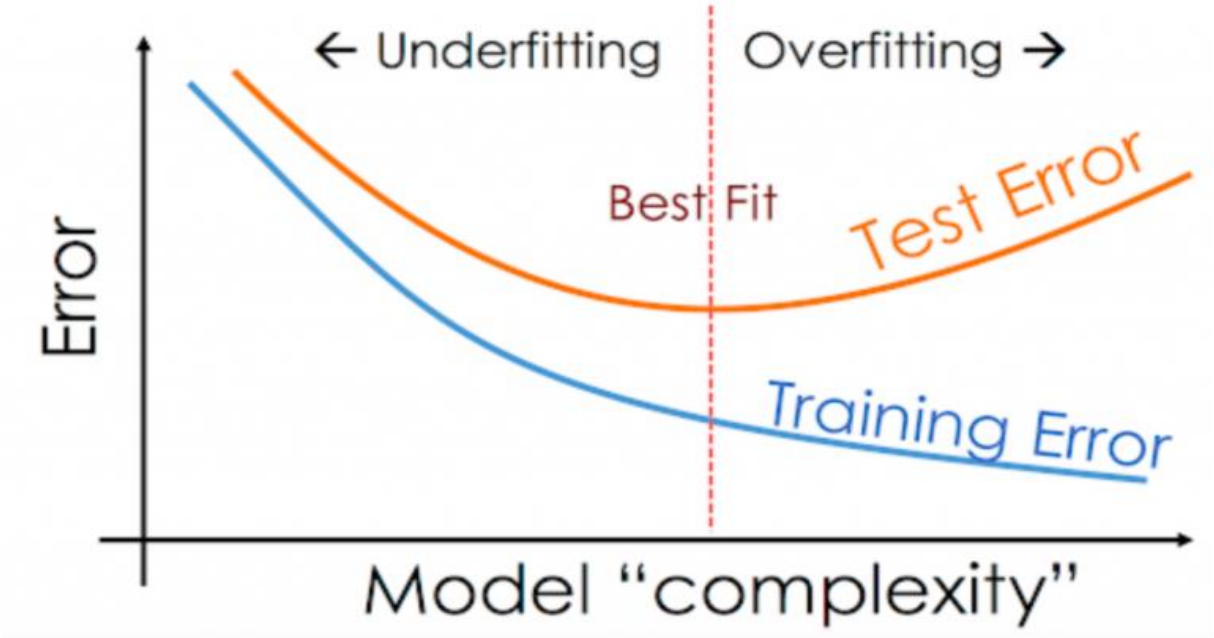


Figure2-8-représentation graphique d'underfitting et overfitting**[f15].**

7-1-Les symptômes de l'underfitting:

Voici les symptômes essentiels de l'overfitting du CNN :

- Performances médiocres sur l'ensemble d'entraînement et de validation/test.
- Courbe d'apprentissage qui n'arrive pas à converger**.[w03]**

7-2-Les causes de l'underfitting dans les CNN:

- Modèle trop simple, avec trop peu de couches ou de paramètres.
- Hyperparamètres mal ajustés (taux d'apprentissage trop faible, etc.).
- Problèmes de convergence durant l'entraînement.

Pour résoudre ces problèmes, il faut trouver le bon équilibre entre la complexité du modèle et la quantité de données disponibles. Les techniques de régularisation jouent un rôle crucial pour éviter l'overfitting, tandis que l'augmentation de la capacité du modèle (plus de couches de filtres, etc.) permet de réduire l'underfitting.

8- La régularisation du CNN :

La régularisation est un ensemble de techniques visant à prévenir le surapprentissage (overfitting) des modèles de deep learning, tels que les CNN. Le surapprentissage se produit lorsque le modèle s'ajuste trop précisément aux données d'entraînement, perdant ainsi sa capacité de généralisation sur de nouvelles données. La régularisation permet de trouver un équilibre entre la complexité du modèle et sa capacité de généralisation.**[05]**

8-1-Le Dropout:

Le dropout consiste à désactiver aléatoirement certaines unités (neurones) du réseau pendant l'entraînement.

Cela force le modèle à apprendre des représentations plus robustes et empêche les neurones de trop dépendre les uns des autres. **[05]**

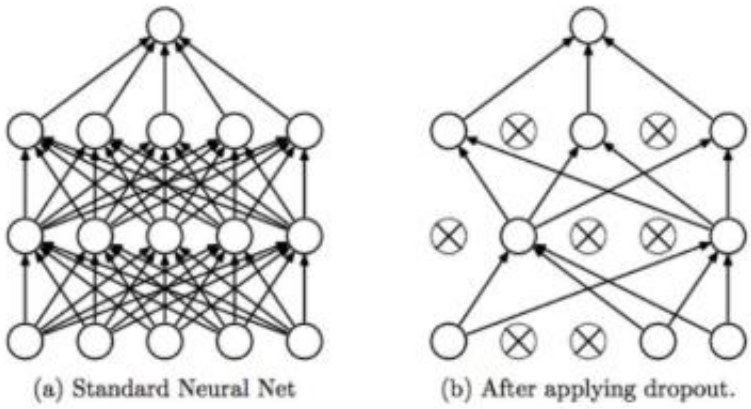


Figure 2-9-Réseau neuronal standard VS Après l'application de Dropout**[f10]**

8-2- L'augmentation de données :

- L'augmentation de données consiste à générer de nouvelles données d'entraînement à partir des données existantes (rotation, translation, zoom, etc.).
- Cela étend artificiellement la taille de l'ensemble d'entraînement et améliore la capacité de généralisation du modèle**.[w03]**

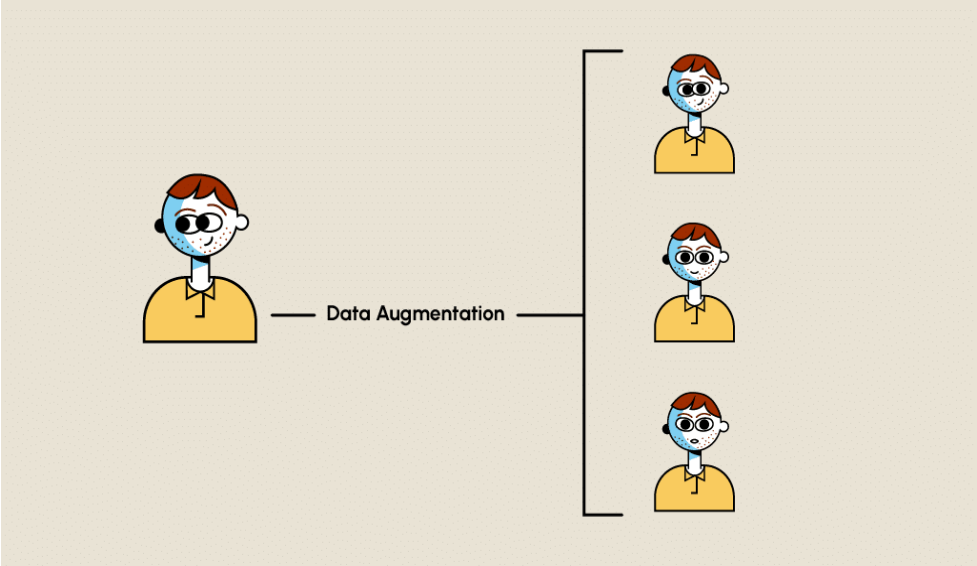


Figure2-10-Exemple d'une image augmentée**[f09]**

8-3- L'Early Stopping :

- L'early stopping consiste à arrêter l'entraînement lorsque les performances sur un ensemble de validation cessent de s'améliorer.
- Cela évite le surapprentissage en stoppant l'entraînement avant que le modèle ne devienne trop complexe**.[w03]**

9-Les variantes les plus populaires de l'architecture CNN :

Les architectures CNN de référence ont servi de base pour de nombreuses variantes et améliorations qui ont permis d'importants progrès en vision par ordinateur ces dernières années. Les principaux réseaux CNN classiques sont LeNet, AlexNet, VGG24, GoogLeNet25, Resnet26, DenseNet27, etc. L'ordre de l'évolution de LeNet à DenseNet est illustré à la figure suivante.

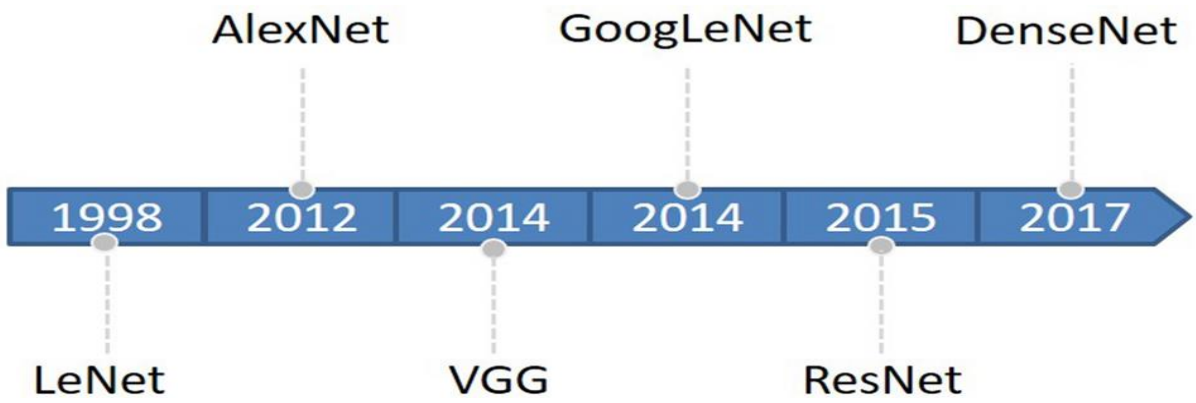


Figure2-11- Chronologie de l'évolution des CNN, de LeNet à DenseNet.

9-1-Exemple de variante du CNN :

9-1-1-AlexNet (2012):

- Architecture profonde révolutionnaire avec 8 couches
- A introduit des innovations comme le ReLU et le dropout
- A établi un nouveau standard de performance sur ImageNet

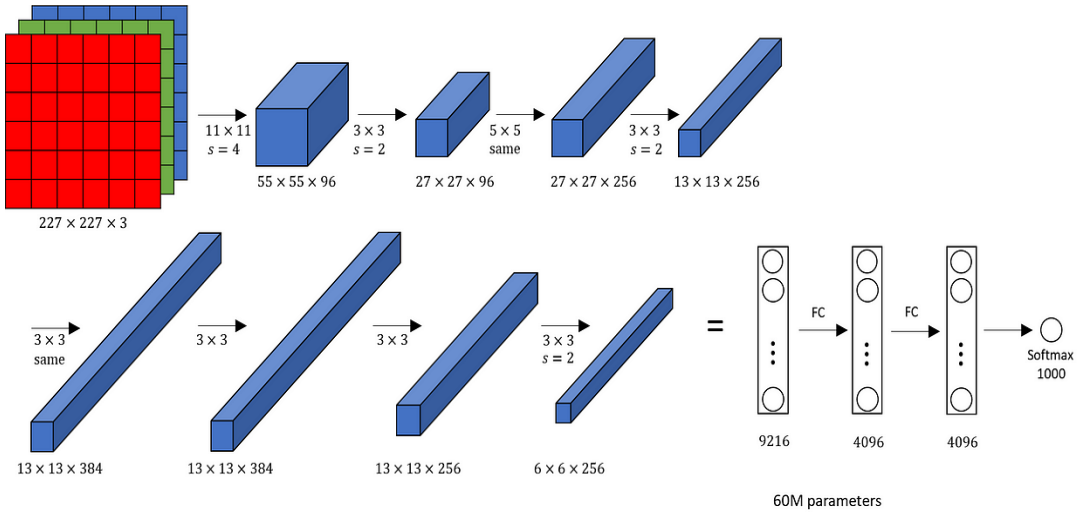


Figure2-12 :l'architecture d'**AlexNet**

9-1-2-ResNet (2015) :

ResNet (Residual Network) est une architecture de réseau de neurones profond proposée par Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun en 2015. Son but était de résoudre le problème du vanishing/exploding gradient qui peut survenir dans les réseaux de neurones très profonds.

L'idée clé derrière ResNet est l'utilisation de connexions résiduelles, également connues sous le nom de connexions par saut. Au lieu d'apprendre la cartographie directe entre l'entrée et la sortie, le réseau apprend la cartographie résiduelle. Cela signifie que le réseau apprend la différence entre la sortie désirée et la cartographie d'identité, ce qui peut être plus facile à apprendre.

Le bloc de construction de base d'un ResNet est un bloc résiduel, qui se compose de deux couches de convolution ou plus, avec une connexion de raccourci qui ignore une ou plusieurs couches. La connexion de raccourci effectue une cartographie d'identité, et sa sortie est ajoutée à la sortie des couches empilées.**[w15]**

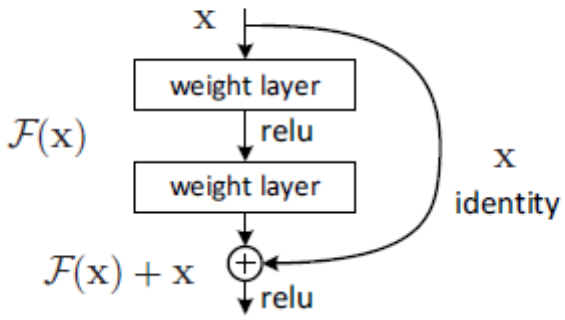


Figure2-13-Bloc de construction résiduel.**[f20]**

Cette architecture a les caractéristiques suivantes :

- Architecture très profonde (jusqu'à 152 couches)
- Utilise des connexions résiduelles pour faciliter l'entraînement
- A établi un nouveau record de performance sur ImageNet

9-1-2-1-Les variantes de l'architecture ResNet :

ResNet ce présente sous différentes versions : ResNet-18/34/50/101/152 :

- Ce sont les versions originales de ResNet, avec un nombre de couches de 18, 34, 50, 101 et 152 respectivement.
- Plus le nombre de couches est élevé, plus le modèle a de capacité de représentation, mais aussi un coût de calcul plus important. **.[w15]**

10- Difficulté de la mise en place d'un nouveau réseau de neurones convolutifs (from scratch):

Créer un nouveau réseau de neurones convolutif est coûteux en terme d'expertise, de matériel et de quantité de données annotées nécessaires.

Il s'agit d'abord de fixer l'architecture du réseau, c'est-à-dire le nombre de couches, leurs tailles et les opérations matricielles qui les connectent. L'entraînement consiste alors à optimiser les coefficients du réseau pour minimiser l'erreur de classification en sortie. Cet entraînement peut prendre plusieurs semaines pour les meilleurs CNN, avec de nombreux GPU travaillant sur des centaines de milliers d'images annotées.

L'apprentissage d'un nouveau modèle de vision par ordinateur basé sur les réseaux de neurones convolutifs représente un défi important. Concevoir et optimiser une telle architecture neuronale complexe nécessite une quantité substantielle de données d'entraînement, de puissance de calcul et de savoir-faire technique.

Les réseaux de neurones convolutifs sont des modèles d'apprentissage profond particulièrement efficaces pour les tâches de classification et de reconnaissance d'images. Cependant, leur mise en place et leur optimisation sont souvent longues et fastidieuses**.[w03]**

Il faut trouver la bonne combinaison d'hyperparamètres, de couches et de techniques de régularisation pour obtenir des performances satisfaisantes.

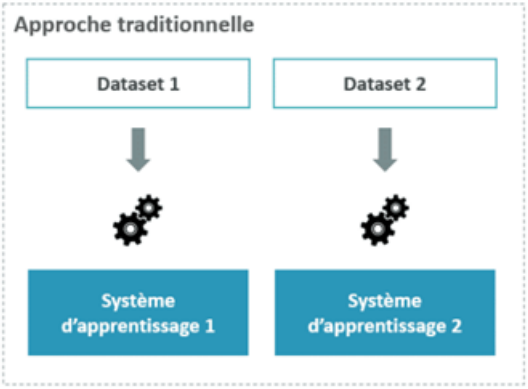


Figure2-14-approche traditionnelle du CNN (from scratch)**.[f09]**

11-Le Transfer Learning (apprentissage par transfert) :

La réutilisation d'un modèle pré-entraîné sur un nouveau problème est connue sous le nom d'apprentissage par transfert dans le domaine de l'apprentissage automatique. Dans le cadre de l'apprentissage par transfert, une machine utilise les connaissances acquises lors d'une tâche antérieure pour améliorer la prédiction d'une nouvelle tâche. **[w16]**

Le Transfer Learning correspond à la capacité à utiliser des connaissances existantes, développées pour la résolution de problématiques données, pour résoudre une nouvelle problématique.

L'apprentissage par transfert est un problème de recherche dans le domaine de l'apprentissage automatique qui se concentre sur le stockage des connaissances acquises lors de la résolution d'un problème et sur leur application à un problème différent mais connexe.

Pour des usages pratiques, il est possible d'exploiter la puissance des CNN sans être un expert du domaine, avec du matériel accessible et une quantité raisonnable de données . Toute la complexité de création de CNN peut être évitée en adaptant des réseaux pré-entraînés disponibles publiquement. Ces techniques sont appelées *transfert learning*, car on exploite la connaissance acquise sur un problème général pour l'appliquer de nouveau à un problème particulier. **[w03]**

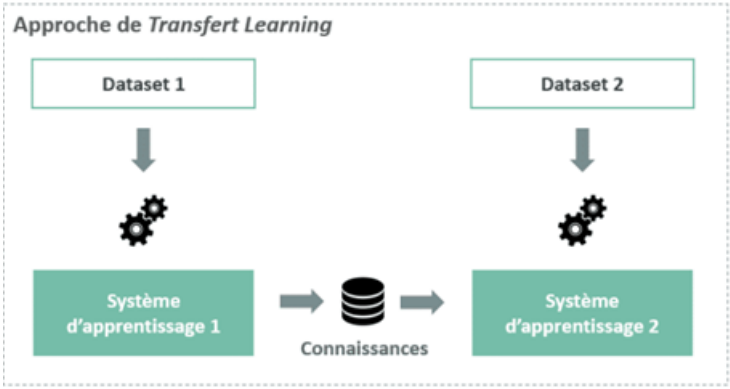


Figure2-15- Approche de Transfert Learning **[f09]**

12-Avantages de transfer learning :

Voici les principaux avantages de l'apprentissage par transfer learning :

1. **Réduction de la quantité de données requise** : Le transfer learning permet d'obtenir de bonnes performances sur une nouvelle tâche avec beaucoup moins de données d'entraînement que si on devait tout apprendre depuis le début. Cela est particulièrement utile quand les données sont coûteuses ou difficiles à obtenir.

2. **Accélération de l'apprentissage** : En réutilisant les représentations apprises sur une tâche source, le temps et les ressources nécessaires pour entraîner un nouveau modèle sont fortement réduits.

3. **Amélioration des performances** : Grâce aux connaissances transférées, les modèles entraînés par transfer learning ont généralement de meilleures performances que ceux entraînés à partir de zéro, surtout lorsque les tâches source et cible sont proches.

4. **Généralisation améliorée** : Les représentations apprises sur une large base de données génériques permettent au modèle de mieux généraliser à de nouvelles données, au-delà de la tâche d'entraînement.

5. **Facilité d'entraînement** : En "finetunant" seulement les dernières couches d'un modèle pré-entraîné, l'entraînement devient plus simple et stable que de devoir entraîner un modèle complet.
6. **Transférabilité des connaissances** : Les features apprises sur une tâche peuvent souvent être réutilisées avec succès sur d'autres tâches similaires, démontrant la transférabilité des connaissances.

13-Exemple de mise en œuvre du transfer learning:

Voici un exemple de mise en œuvre du transfer learning avec le modèle ResNet50 pré-entraîné :

Supposons que nous voulions classer des images de fleurs en plusieurs catégories (roses, tulipes, orchidées, etc.). Nous pouvons utiliser le modèle ResNet50 pré-entraîné sur ImageNet comme point de départ.

Étapes du transfer learning avec ResNet50 :

1. **Chargement du modèle pré-entraîné** : Nous chargeons le modèle ResNet50 pré-entraîné sur ImageNet, disponible par exemple dans la librairie Keras.

2. **Gel des couches de base** : Nous gelons les poids des couches du modèle de base, pour ne pas les modifier pendant l'entraînement sur nos données de fleurs.

3. **Ajout de nouvelles couches** : Nous ajoutons de nouvelles couches en haut du modèle de base pour adapter le modèle à notre tâche de classification de fleurs.

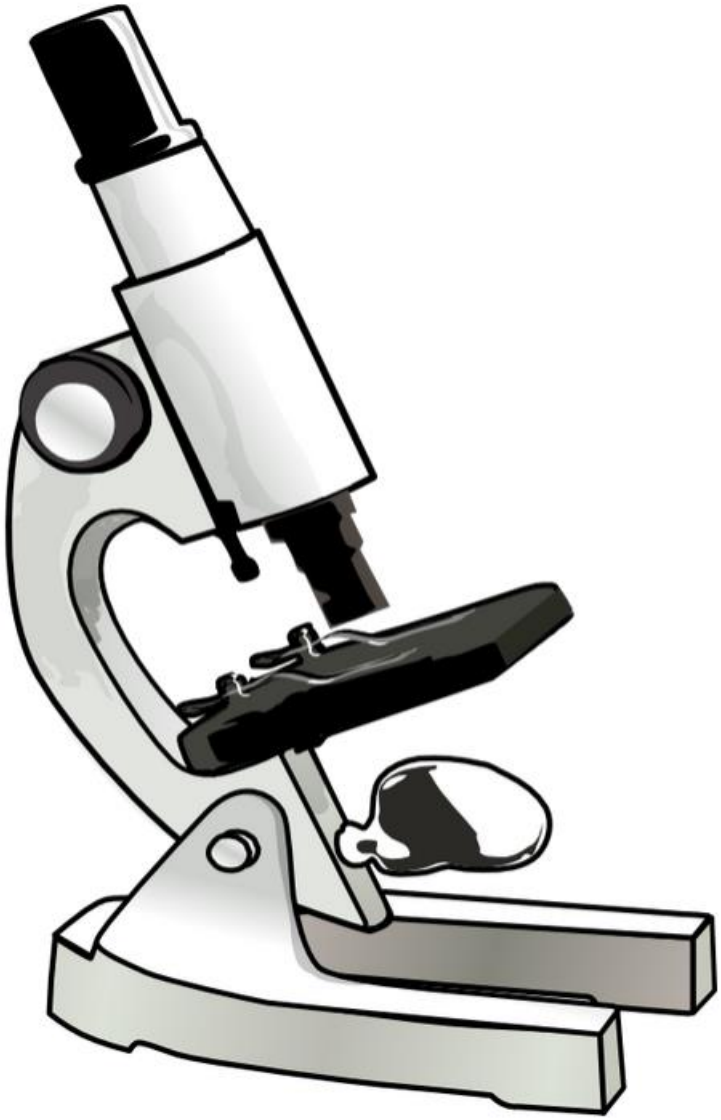
4. **Entraînement du modèle** : Nous entraînons uniquement les nouvelles couches ajoutées, en utilisant les données de fleurs.

Grâce au transfer learning, nous pouvons tirer parti des représentations visuelles génériques apprises par ResNet50 sur ImageNet, et nous concentrer uniquement sur l'adaptation du modèle à notre tâche spécifique de classification de fleurs. Cela permet d'obtenir de meilleures performances avec moins de données d'entraînement.

14-Conclusion :

Ce chapitre nous a permis d'acquérir une compréhension approfondie des réseaux de neurones convolutifs (CNN), une famille de modèles d'apprentissage profond particulièrement adaptés aux tâches de traitement d'images. Nous avons examiné leur structure ainsi que leur fonctionnement qui permet d'extraire progressivement des features de plus en plus complexes à partir des images en entrée , ainsi que Les principales variantes et architectures de CNN, telle que ResNet

Enfin, nous avons abordé les deux principales approches de construction d'un réseau CNN : l'entraînement depuis zéro ("from scratch") et le transfert d'apprentissage ("transfer learning"). Ces dernières méthodes que nous allons l'exploiter dans le prochain chapitre pour construire notre propre système d'annotation du langage ASL.



Chapitre03 :

Conception

1-Introduction :

Après avoir présenté les fondements des systèmes de La langue des signes américaine (American Sign Language en anglais) et de l'intelligence artificielle dans le premier chapitre, et avoir approfondi les détails des réseaux neuronaux convolutifs dans le deuxième chapitre,

Nous allons proposer dans cette partie, deux approches permettant d'aider à la mise en œuvre de deux modèles d'annotation d'image, d'une façon très synthétique,

Ce chapitre se concentre sur la conception de nos modèles et les détaille .

2-Logique suivie :

Puisque le CNN est un réseau adaptatif, nous devons l'entraîner à s'adapter à n'importe quelle entrée donnée.

Pour l'entraînement, un ensemble d'images (American Sign Language (ASL) Alphabet)sont collectées à partir de notre Dataset (ASL Alphabet).

Afin de donner une image comme entrée à ce CNN, nous devons rendre cette image dans un format particulier, c'est-à-dire, nous devons la compresser comme un fichier JPEG. Plus de nombre d'images utilisées pour la formation est grand, plus grande sera l'efficacité du CNN (ici on a 87000 images au format JPEG).

Après cela, le CNN est prêt pour l'annotation. Maintenant l'image de signe peut être donnée comme entrée au CNN et nous obtiendrons l'annotation correspondante en tant que sortie.

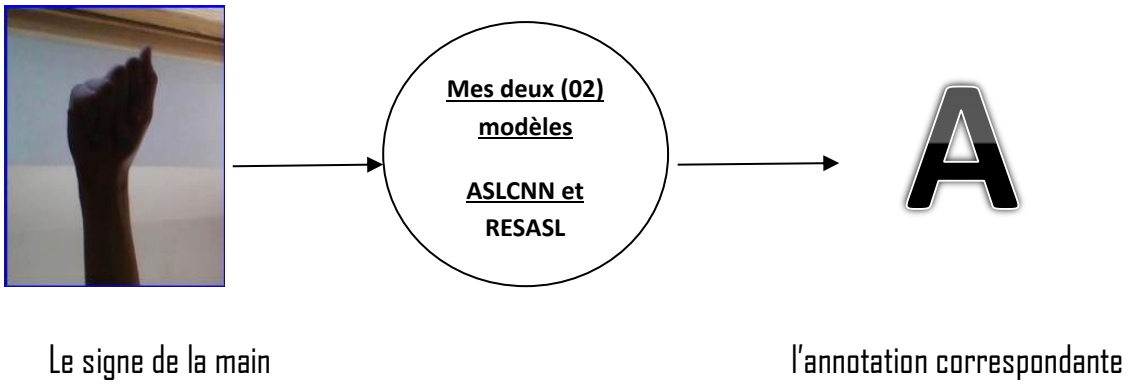


Figure3-1- Fonctionnement global de mes modèles.

3-Conception globale de mes modèles:

Dans le présent mémoire j'ai opté d'utiliser deux architectures différentes de CNN (RESASL par l'apprentissage par transfert et ASLCNN un modèle construit à partir de zéro) pour prédire la langue des signes des images de l'ASL.

Cette conception a la structure globale suivante :

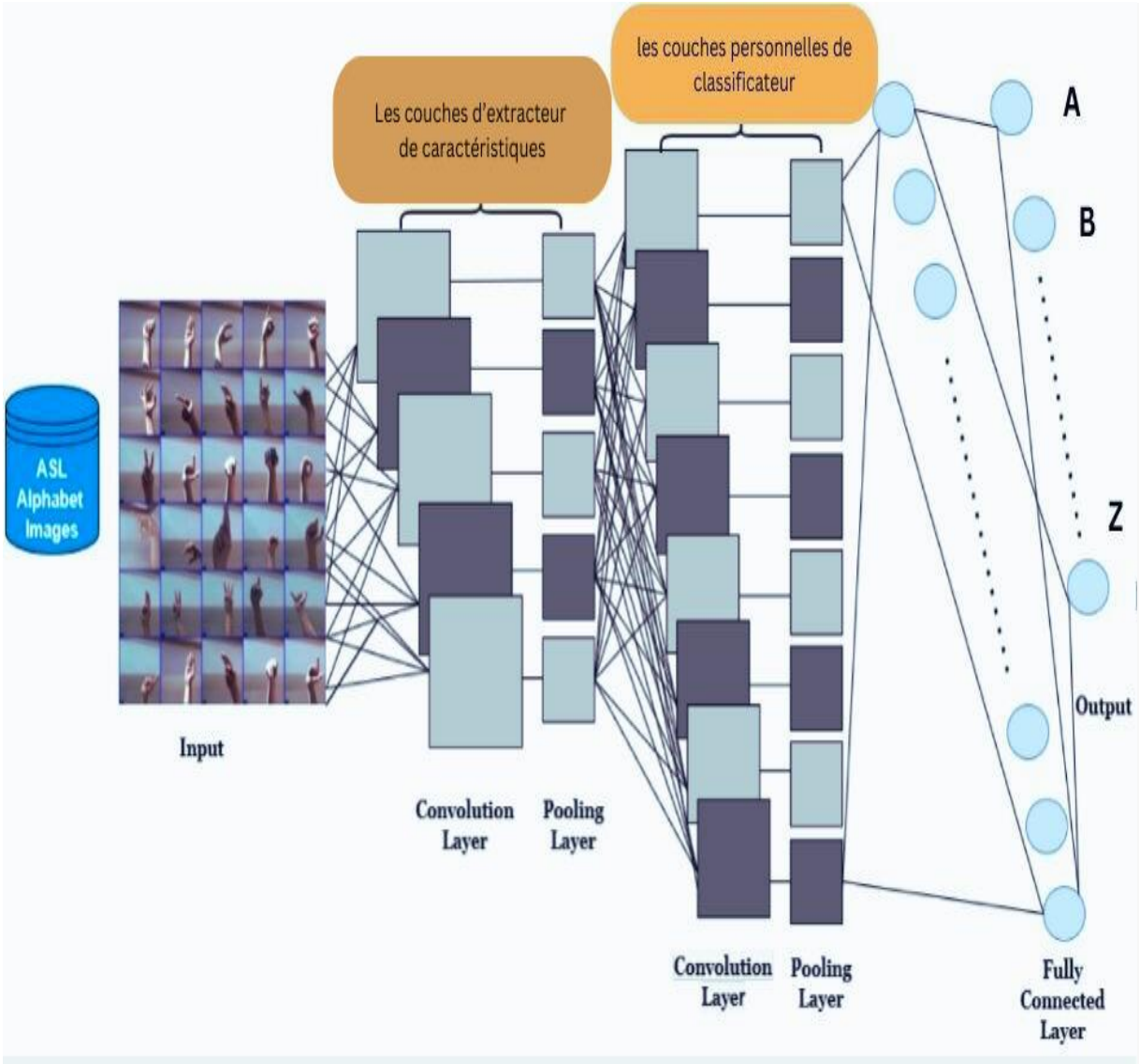


Figure3-2- Notre structure globale du système d'annotation

3-1-La première conception (le modèle 01 ASLCNN): (Un CNN from scratch) :

Dans ce travail, nous essayons d'atteindre cet objectif en utilisant la capacité d'apprentissage adaptatif d'un CNN en construisant ce modèle à partir du zéro (from scratch) pour le faire on a utilisé la structure suivante ;

3-1-1-La structure utilisée dans le modèle 01 (ASLCNN):

Pour cette première implémentation on a utilisé une structure basée sur Le réseau neuronal convolutif composé de plusieurs blocs de couches, dont des couches convolutives, des couches de normalisation, de pooling , flatten, dropout et des couches d'activation comme indiqué dans l'image suivante :

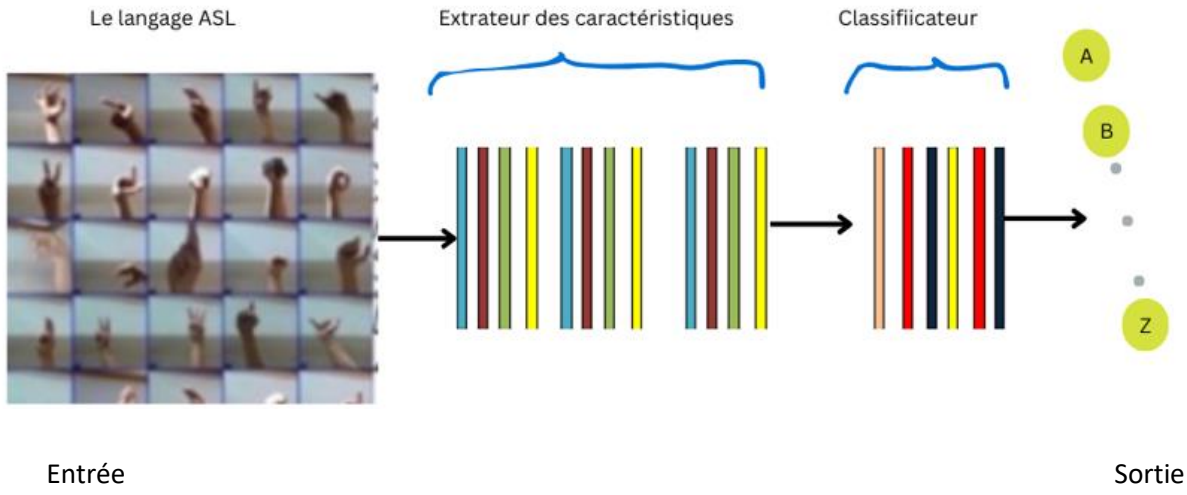


Figure3-3-La structure du système proposé ASLCNN.

Couche convolution Batch Normalisation Maxpooling Dropout

Flatten Dense Activation

Le réseau est constitué d'une chaîne de couches convolutionnelles symétriques. Les couches convolutionnelles agissent comme des extracteurs de caractéristiques, elles permettent d'extraire efficacement les informations visuelles les plus importantes à partir des signes de main.

le max pooling est une opération essentielle dans les CNN car elle permet de réduire la dimensionnalité tout en conservant les caractéristiques les plus discriminantes. Cela améliore l'efficacité du modèle et sa capacité de généralisation.

la batch normalization est une technique puissante qui améliore considérablement la stabilité, la convergence et la généralisation des réseaux de neurones profonds.

Enfin j'ai utilisé le dropout comme technique de régularisation essentielle qui améliore grandement la généralisation des réseaux de neurones profonds en prévenant le sur-apprentissage de mon modèle.

Dont le détail de cette structure est le suivant (sachant que ce schéma est extrait par le biais de l'instruction

keras.utils.plot_model

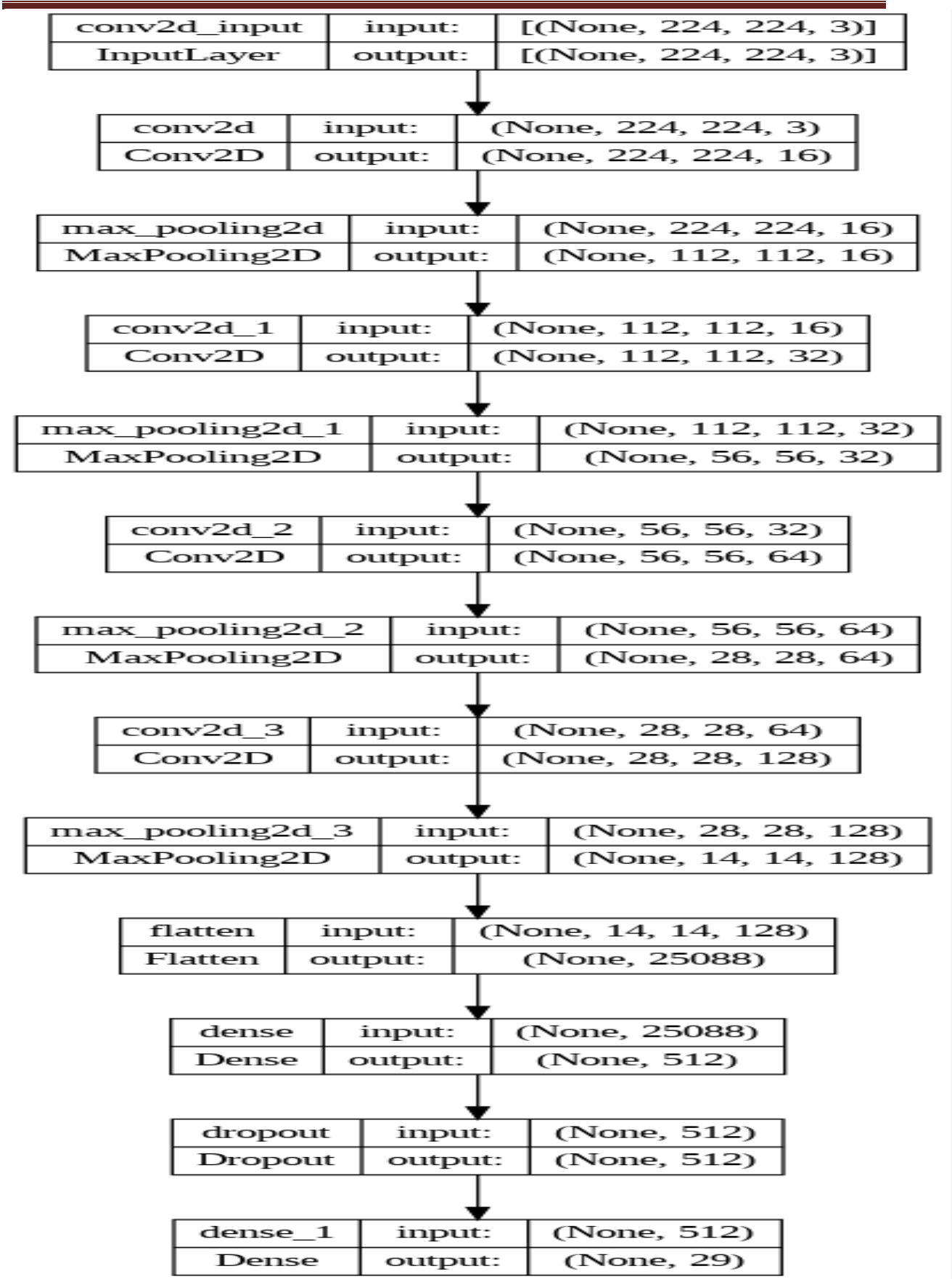


Figure3-4-La structure détaillée du système proposé ASLCNN

3-1-2-L'algorithme proposé du modèle 01 (ASLCNN):

Notre objectif est de générer une annotation à partir du signe de la main.

Notre modèle travaille selon l'algorithmique suivant :

Algorithme global du ASLCNN

1-Introduire les images RGB de taille 224X224.

Nombre_filtre=32

2-Pour i de 1 à 3 (3 blocs) faire :

1-Appliquer les Convolutions sur l'image y avec Nombre_filtre de taille 5x5 avec une fonction d'activation.

2-Appliquer une normalisation des lots (BatchNormalization) pour réduire considérablement le nombre d'époques de formation nécessaires pour former des réseaux profonds..

3-Appliquer un maxpooling max pooling 2D avec une fenêtre de taille 4x4

4-Ajouter une couche de dropout de 25% à notre modèle.

5- Nombre_filtre= Nombre_filtre X 2

Fin pour

3- Ajouter d'une couche "Flatten"**.**

4- Ajouter d'une couche dense (fully connected) avec 512 neurones et une fonction d'activation ReLU

5-Appliquer une normalisation des lots (BatchNormalization).

6-Ajouter une couche de dropout de 50% à notre modèle.

7- Ajouter une couche dense finale de 29 classes de prédiction avec une activation softmax

Retourner l'annotation correspondante.

3-2-La deuxième conception (le modèle 02 RESASL): (by transfer learning):

Cette deuxième implémentation n'est que la première implémentation(ASCNN) en lui remplaçant ses couches d'extraction de caractéristiques par l'extracteur du RESNET50, à travers l'apprentissage par transfert.

3-2-1-La structure utilisée dans le modèle 02 (RESASL):

Comme la plupart des réseaux, le réseau RESASL est composé de deux éléments :

- un extracteur de caractéristiques
- Un classificateur

Pour cette deuxième implémentation on a modifié la première implémentation on lui remplaçant son extracteur de caractéristiques par l'extracteur RESNET50 pré-entrainé sur le gigantesque Dataset ImageNet ce qui nous permet d'utiliser les connaissances acquises sur cette base contenant plus d'un million d'images pour résoudre notre problème ayant des données plus limitées (87000 images) tout en gardant les mêmes couches du classificateur de la première implémentation ASLCNN pour effectuer une classification spécifique à un problème avec 29 classes , comme le montre la figure suivante.

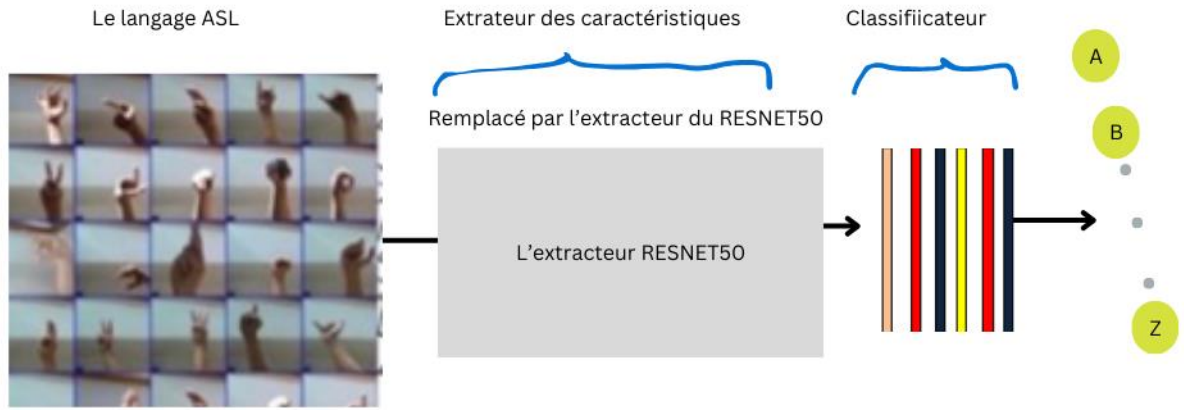


Figure3-5-La structure du système proposé RESASL.

Dropout Flatten

Dense Activation

3-2-2-L'algorithme proposé du modèle RESASL :

Notre objectif objectif est de générer une annotation à partir du signe de la main.

Notre modèle travaille selon l'algorithmique suivant :

Algorithme global

1. **Chargement du modèle de base ResNet50**:
   - Chargement le modèle ResNet50 pré-entraîné sur la base de données ImageNet, en excluant la couche de classification finale (avec include_top=False) et en spécifiant la taille d'entrée 224x224 pixels avec 3 canaux de couleur.

2. **Extraction des caractéristiques du modèle de base**:
   - input = baseModel.layers[0].input : On récupère l'entrée du modèle de base.
   - output = baseModel.output : On récupère la sortie du modèle de base, qui correspond aux caractéristiques extraites.

3. **Ajout des couches du classificateur**:

   3-1- Ajouter d'une couche "Flatten"**.**

   3-2- Ajouter d'une couche dense (fully connected) avec 128 neurones

   et une fonction d'activation ReLU.

   3-3-Appliquer une normalisation des lots (BatchNormalization).

   3-4-Ajouter une couche de dropout de 50% à notre modèle.

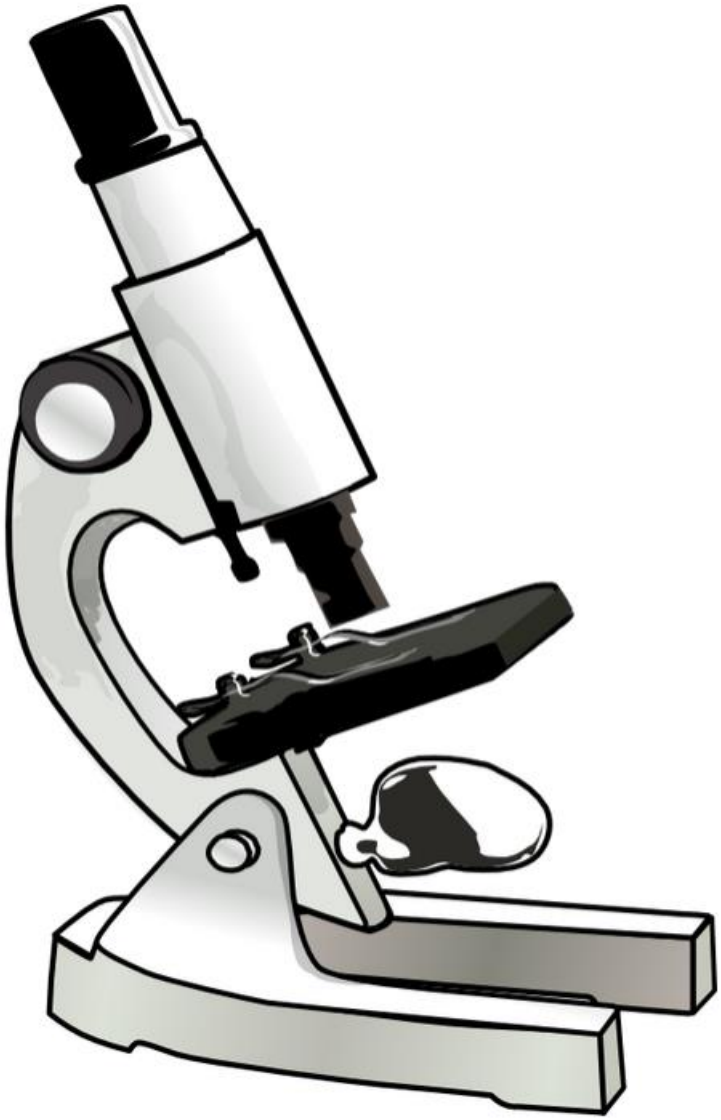
   3-5-Ajouter une couche dense finale de 29 classes de prédiction

   avec une activation softmax

4. **Construction du modèle final**:
   - model = keras.Model(inputs=input, outputs=fcHead) : On construit le modèle final en reliant l'entrée du modèle de base aux couches finales ajoutées.

4-Conclusion :

Après avoir conçu nos systèmes (basé sur une conception CNN from scratch et une conception par le transfer learning) au cours de ce chapitres, les prochaines étapes consistent à implémenter et estimer les performances des modèles, à effectuer des tests de performance approfondis, ainsi qu'à explorer en détail les résultats expérimentaux obtenus.



Chapitre04 :

Implémentation et évaluation

1-Introduction :

Après avoir conçu mes modèles, il est maintenant temps de les implémenter en utilisant le matériels et les outils appropriés ainsi que les évaluer de manière approfondie pour l'estimation des performances du modèle, où je définirai des métriques pertinentes, mettrai en place des jeux de test, et analyserai en détail les résultats obtenus.

2-Le matériel utilisé:

Pour réaliser notre application on a utilisé un micro portable qui a les spécifications suivantes :

Marque	DELL
Modèle	Dell Latitude 5480
Processeur	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz
RAM	8,0 Go
Système utilisé	Windows 10 Professionnel
Cartes Graphiques	Cartes graphiques Intel(R) HD Graphics 520

Tableau4-1-les spécificités du matériel utilisé

Cependant, en raison de la taille importante du dataset (87 000 images) et des calculs associés, les spécifications de mon ordinateur ne sont pas suffisantes pour réaliser ce modèle.

Par conséquent, je suis contraint d'utiliser une machine virtuelle dans le cloud (Google) qui offre deux options : une option gratuite utilisant uniquement le CPU pour les calculs, et une option payante offrant une quantité importante de mémoire et un GPU dédié aux calculs graphiques et matriciels.

3- Présentation des outils de développement :

Voici quelques outils constituent une base solide pour développer des applications d'apprentissage profond performantes et évolutives dans divers domaines, notamment la vision par ordinateur, le traitement du langage naturel et la prédiction de séries temporelles.

3-1- Python :

Python est un langage de programmation de haut niveau interprété, orienté objet et doté d'une sémantique dynamique. Ses structures de données de haut niveau, combinées au typage dynamique et à la liaison dynamique, le rendent très intéressant pour le développement rapide d'applications, ainsi que pour une utilisation en tant que langage de script ou de colle pour relier des composants existants entre eux. La syntaxe simple et facile à apprendre de Python met l'accent sur la lisibilité et réduit donc le coût de la maintenance des programmes. Python

prend en charge les modules et les paquets, ce qui favorise la modularité des programmes et la réutilisation du code. L'interpréteur Python et la vaste bibliothèque standard sont disponibles gratuitement sous forme de source ou de binaire pour toutes les principales plates-formes et peuvent être distribués librement.

Figure4-1-logo du langage python **[f17]**

3-2- TensorFlow :

Il s'agit d'une bibliothèque de mathématiques symboliques qui utilise le flux de données et la programmation différentiable pour effectuer diverses tâches axées sur la formation et l'inférence de réseaux neuronaux profonds.

Son architecture flexible permet un déploiement facile des calculs sur une variété de plateformes (CPU, GPU, TPU), et des ordinateurs de bureau aux clusters de serveurs en passant par les appareils mobiles et périphériques.

Développé à l'origine par des chercheurs et des ingénieurs de l'équipe Google Brain au sein de l'organisation IA de Google, il offre un support solide pour l'apprentissage automatique et l'apprentissage profond.

Figure4-2-logo de la bibliothèque TensorFlow **[f18]**

3-3- Keras :

Keras est l'API de haut niveau de la plateforme TensorFlow. Elle fournit une interface accessible et hautement productive pour résoudre les problèmes d'apprentissage machine (ML), en mettant l'accent sur l'apprentissage profond moderne. Keras couvre toutes les étapes du flux de travail de l'apprentissage automatique, du traitement des données au déploiement en passant par le réglage des hyperparamètres. Il a été développé pour permettre une expérimentation rapide.

Figure4-3-logo de l'API Keras **[f19]**.

3-4-Google Colaboratory :

Egalement connu sous le nom de Colab, est une plateforme de notebook Jupyter gratuite qui fonctionne entièrement dans le cloud. Elle offre un environnement de développement sans nécessiter de configuration préalable. Les notebooks que vous créez sur Colab peuvent être édités simultanément par les membres de votre équipe, de la même manière que vous éditez des documents sur Google Docs.
Une caractéristique importante de Google Colab est qu'il prend en charge de nombreuses bibliothèques d'apprentissage automatique populaires. Ces bibliothèques peuvent être facilement chargées dans votre notebook, ce qui facilite l'utilisation de fonctionnalités avancées d'apprentissage automatique.
De plus, Google Colab offre la possibilité d'exécuter vos notebooks sur des GPU, ce qui évite les limitations de ressources rencontrées avec des machines à faible puissance de calcul. Cela permet d'accélérer considérablement les calculs, en particulier pour les tâches intensives en termes de graphiques et de calculs matriciels liées à l'apprentissage automatique.

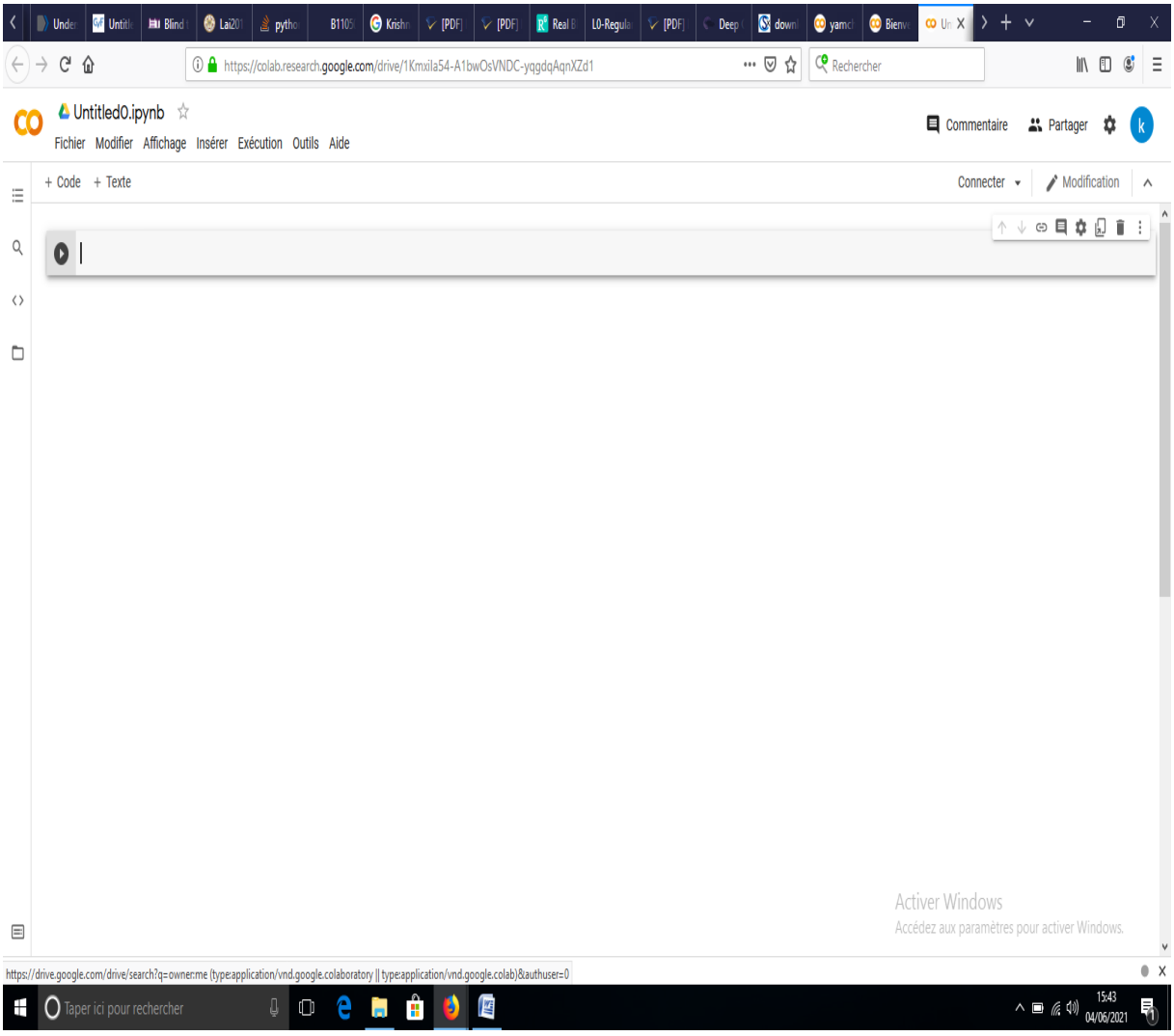


Figure4-4-: Interface du Google Colaboratory

Remarque:

Pour développer et entrainer notre modèle on a choisi l'environnement de Google qui a fourni l'environnement de programmation Google Colab, gratuit (utilisation limitée en terme de GPU) et Pro dont lequel j'ai acheté 500 (5 X 100) unités pour mieux accélérer l'exécution de notre modèle, sachant qu'on a essayé vainement sur la version gratuite qui a introduit un plantage de la machine virtuelle vu la taille imposante de données traitées.

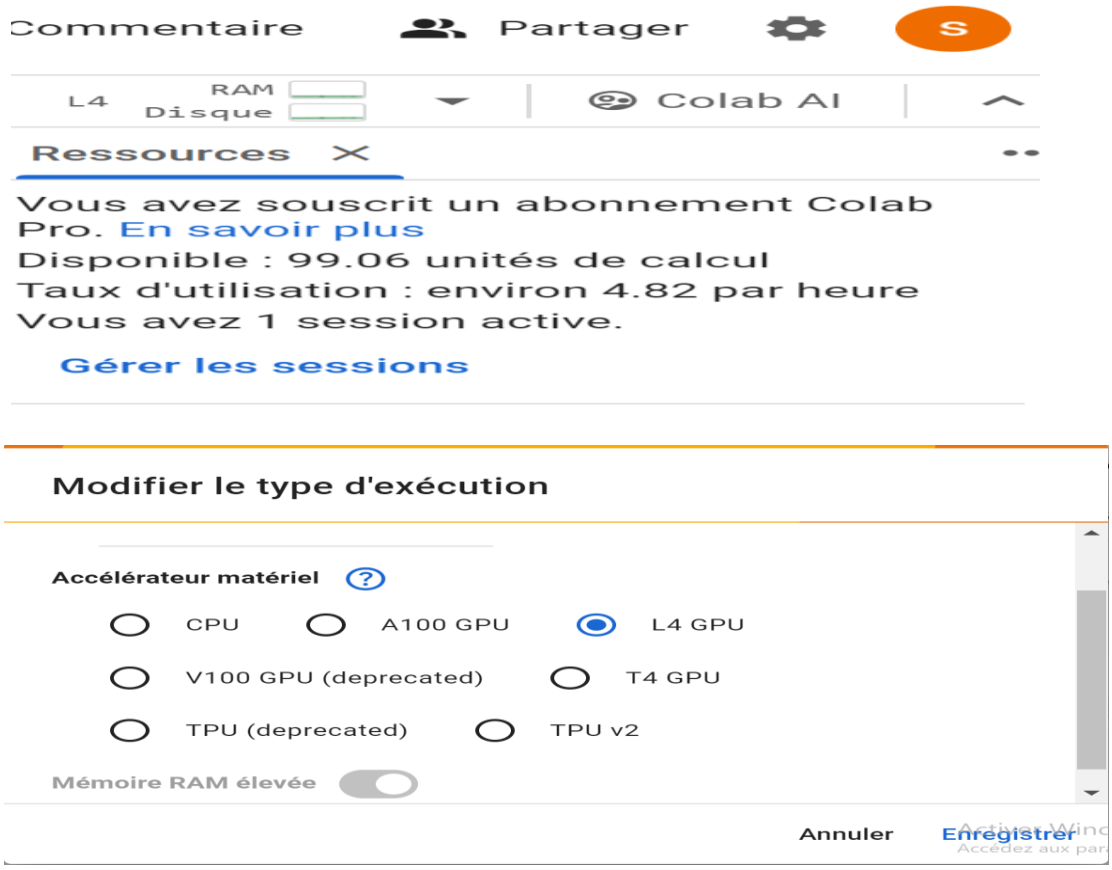


Figure4-5-choix du type d'exécution dans Google Colaboratory

Sachant que l'entrainement sur un GPU est extrêmement rapide qu'un entrainement classique sur un CPU.

4-Le dataset utilisé :

L'ensemble des données d'apprentissage utilisé est un dataset téléchargé librement du site kaggle contenant 87 000 images de 200x200 pixels d'une taille globale de (1.02 Go) dans lequel Il y a 29 classes, dont 26 pour les lettres de A à Z et 3 classes pour ESPACE, SUPPRIMER et RIEN, ces trois classes sont très utiles pour les applications en temps réel et la classification.

L'ensemble de données de test ne contient que 29 images, afin d'encourager l'utilisation d'images de test réelles.

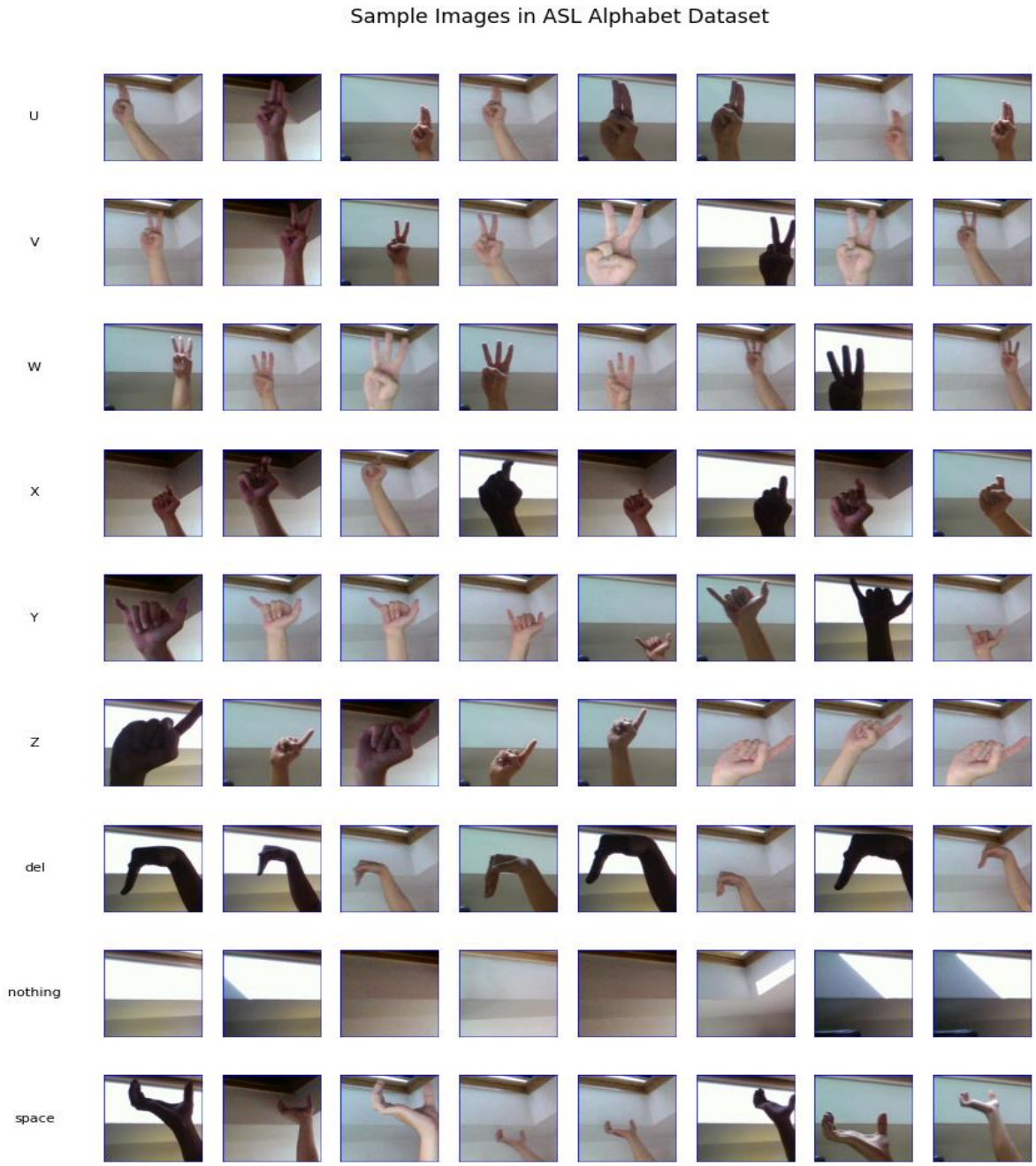


Figure4-6- échantillon d'image utilisé dans notre modèle**[f16]**

5-Préparation (Prétraitement et augmentation) des images du Dataset:

- Redimensionner les images de notre Dataset de dimension d'origine de 200 x200 à la dimension 224x224.

  A travers le code suivant :

```
input_shape=(224, 224, 3)
```

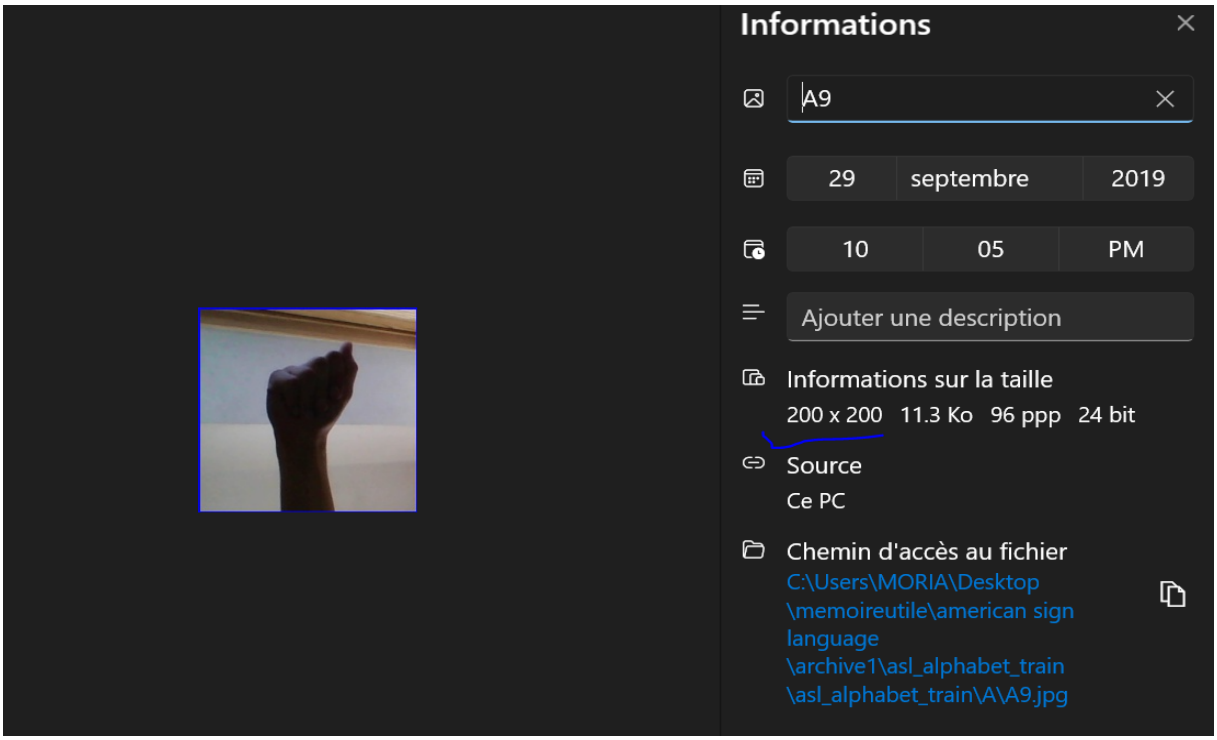


Figure4-7- taille de l'image initiale avant son changement

Pour l'augmentation on a

- Pour chaque pixel, la valeur RGB est divisée par 255.
- Mappage des valeurs RGB [0-255] à la gamme [0-1].

L'augmentation est faite à travers la ligne du code principale suivante :

```
aug = ImageDataGenerator(rescale=1/255.0)
```

6-L'algorithme d'optimisation utilisé:

Pour entrainer les hyper paramètres de notre modèle on utilise la back propagation pour mettre à jour la valeur de chaque hyper paramètre. Pour que la perte (loss) converge plus rapidement, on a utilisé Adam (Adoptive Moment Estimation).

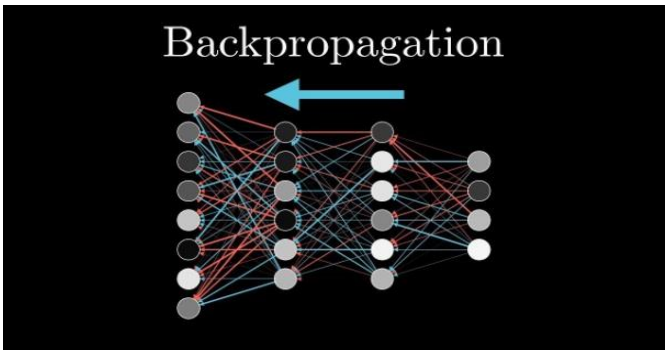


Figure4-8-Fonctionnement de la Backpropagation**[f16]**

7-La fonction de coût utilisée :

La fonction "categorical_crossentropy" (entropie croisée catégorique) est une fonction de perte couramment utilisée dans les tâches de classification multiclasse. Elle est souvent utilisée en conjonction avec la fonction d'activation "softmax" pour calculer la perte entre les prédictions du modèle et les vérités terrain (étiquettes) correspondantes.
La fonction categorical_crossentrop (01) mesure la divergence entre la distribution de probabilité prédite par le modèle et la distribution de probabilité réelle des étiquettes sachant qu'elle a la formule suivante ;

$$loss = -\Sigma(y\_true * log(y\_pred)) \quad ---(01)$$

8-Les démarches de réalisation de l'application :

Pour réaliser cette application on a suivi les étapes suivantes :

1-Importer le Dataset (ASL Alphabet from Kaggle) .

2-Développer deux modèles (nommés ASLCNN et RESASL) aux paramètres aléatoires.

3-Compilation de chaque modèle.

3- Développer un algorithme d'apprentissage pour trouver les paramètres du modèle qui minimisent la fonction coût pour chaque modèle.

4-Développer une fonction coût qui mesure l'erreur entre mes modèles et le Dataset.

5-Mesurer la précision (accuracy) des modèles proposés.

6-Evaluer chaque modèle.

9-Notre première implémentation du modèle 01 (ASLCNN) :

9-1-Codification du modèle 01 (ASLCNN) :

On a tout entre nos mains que ce soit les outils matériels ou logiciels pour implémenter notre système d'annotation dont on a choisi d'utiliser le puissant langage de programmation python; voici une portion du code utilisé dans l'implémentation du modèle 01.

```
    for name in train_paths:
      shutil.copy(name, root_folder + "/train/" +  label )


    for name in val_paths:
      shutil.copy(name, root_folder + "/val/" +  label )


    for name in test_paths:
      shutil.copy(name, root_folder + "/test/" +  label )


  return root_folder + "/train" , root_folder + "/val" ,root_folder + "/test"
```

```
[ ] root_folder = "/content/drive/MyDrive/originsecc/asl_alphabet_train/asl_alphabet_train"
    image_paths = list(paths.list_images(root_folder))



    list_labels =  os.listdir(root_folder)
    print(len(image_paths))
    print((list_labels))
```

```
87000
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'del'
```

Figure4-9-Une portion du code de mon modèle ASLCNN

9-2-Entraînement du modèle 01 (ASLCNN) :

En bref, entraînez le modèle sur les données d'entraînement et validez-le sur les données de test, une fois que nous sommes satisfaits des performances du modèle sur l'ensemble de validation, nous pouvons l'utiliser pour faire des prédictions sur de nouvelles données inconnues.

Dans cette étape, le nombre d'époques est défini, nous avons essayé plusieurs choix et nous avons opté pour 10 époques (les époques sont le nombre de fois que l'apprentissage du modèle est répété).

Le choix de nombre d'époque se fait généralement d'une manière stochastique.

```
Epoch 1/10
476/476 [==============================] - ETA: 0s - loss: 1.6294 -
accuracy: 0.5318
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 454s 925ms/step - loss: 1.6294 -
accuracy: 0.5318 - val_loss: 7.2551 - val_accuracy: 0.1246 - lr: 0.0010
Epoch 2/10
476/476 [==============================] - ETA: 0s - loss: 0.3522 -
accuracy: 0.8824
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 420s 882ms/step - loss: 0.3522 -
accuracy: 0.8824 - val_loss: 0.1802 - val_accuracy: 0.9434 - lr: 0.0010
Epoch 3/10
476/476 [==============================] - ETA: 0s - loss: 0.1647 -
accuracy: 0.9455
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 406s 853ms/step - loss: 0.1647 -
accuracy: 0.9455 - val_loss: 0.2431 - val_accuracy: 0.9224 - lr: 0.0010
Epoch 4/10
476/476 [==============================] - ETA: 0s - loss: 0.1018 -
accuracy: 0.9658
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 403s 846ms/step - loss: 0.1018 -
accuracy: 0.9658 - val_loss: 0.0813 - val_accuracy: 0.9756 - lr: 0.0010
Epoch 5/10
476/476 [==============================] - ETA: 0s - loss: 0.0788 -
accuracy: 0.9741
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 404s 849ms/step - loss: 0.0788 -
accuracy: 0.9741 - val_loss: 0.7103 - val_accuracy: 0.8263 - lr: 0.0010
Epoch 6/10
476/476 [==============================] - ETA: 0s - loss: 0.0665 -
accuracy: 0.9779
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 403s 847ms/step - loss: 0.0665 -
accuracy: 0.9779 - val_loss: 0.9109 - val_accuracy: 0.8160 - lr: 0.0010
Epoch 7/10
476/476 [==============================] - ETA: 0s - loss: 0.0583 -
accuracy: 0.9802
```

```
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 406s 853ms/step - loss: 0.0583 -
accuracy: 0.9802 - val_loss: 0.0812 - val_accuracy: 0.9724 - lr: 0.0010
Epoch 8/10
476/476 [==============================] - ETA: 0s - loss: 0.0500 -
accuracy: 0.9832
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 400s 841ms/step - loss: 0.0500 -
accuracy: 0.9832 - val_loss: 0.0951 - val_accuracy: 0.9656 - lr: 0.0010
Epoch 9/10
476/476 [==============================] - ETA: 0s - loss: 0.0424 -
accuracy: 0.9856
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 404s 848ms/step - loss: 0.0424 -
accuracy: 0.9856 - val_loss: 0.1626 - val_accuracy: 0.9546 - lr: 0.0010
Epoch 10/10
476/476 [==============================] - ETA: 0s - loss: 0.0458 -
accuracy: 0.9850
WARNING:tensorflow:Early stopping conditioned on metric `metrics` which is
not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
476/476 [==============================] - 404s 848ms/step - loss: 0.0458 -
accuracy: 0.9850 - val_loss: 0.0087 - val_accuracy: 0.9969 - lr: 0.0010
```

9-3-Le sauvegarde de notre modèle ASLCNN :

On a sauvegardé notre modèle sous forme d'un fichier avec l'extension ".h5" contenant les poids et la configuration du modèle, ce qui permet de les recharger ultérieurement pour effectuer des prédictions ou poursuivre l'entraînement.

On a sauvegardé notre modèle dans un fichier nommé ASLCNN.h5 par le biais de l'instruction suivante :

model.save('/content/drive/MyDrive/ColabNotebooks/ĐAPM_AI/HandSign/ASLCNN.h5').

l'opération inverse s'appelle le chargement du modèle elle s'effectue comme suit :

```
model=tf.keras.models.load_model("/content/drive/MyDrive/ColabNotebooks/ĐAPM_AI/Hand
Sign/ASLCNN.h5")
```

9-4-Résultats expérimentaux du modèle 01 (ASLCNN):

Nous avons réalisé des expériences en utilisant l'optimiseur Adam pour un taux d'apprentissage (lr=0,001)et un nombre d'epochs=10 pour vérifier l'exactitude du schéma proposé, après l'entrainement la perte pour le réseau se converge vers zéro.

Nous prédisons ensuite les classes pour ces images à l'aide du modèle entraîné et estimons les performances du modèle en comparant le nombre de prédictions correctes et d'erreurs commises.

Figure4-10-Quelques résultats obtenus sur de nouvelles images inconnues.

9-5-Les métriques de mesures de performances de notre modèle 01 (ASLCNN):

L'évaluation des performances de notre modèle est essentielle pour mesurer son efficacité et sa capacité à résoudre la classification. On a utilisé les trois métriques suivantes :

9-5-1-La Précision (Accuracy) du modèle 01 (ASLCNN):

Le modèle a obtenu d'excellents résultats avec une précision près de 99,50 % dans presque toutes les nouvelles images de test inconnues, ce qui est considéré comme un bon résultat.

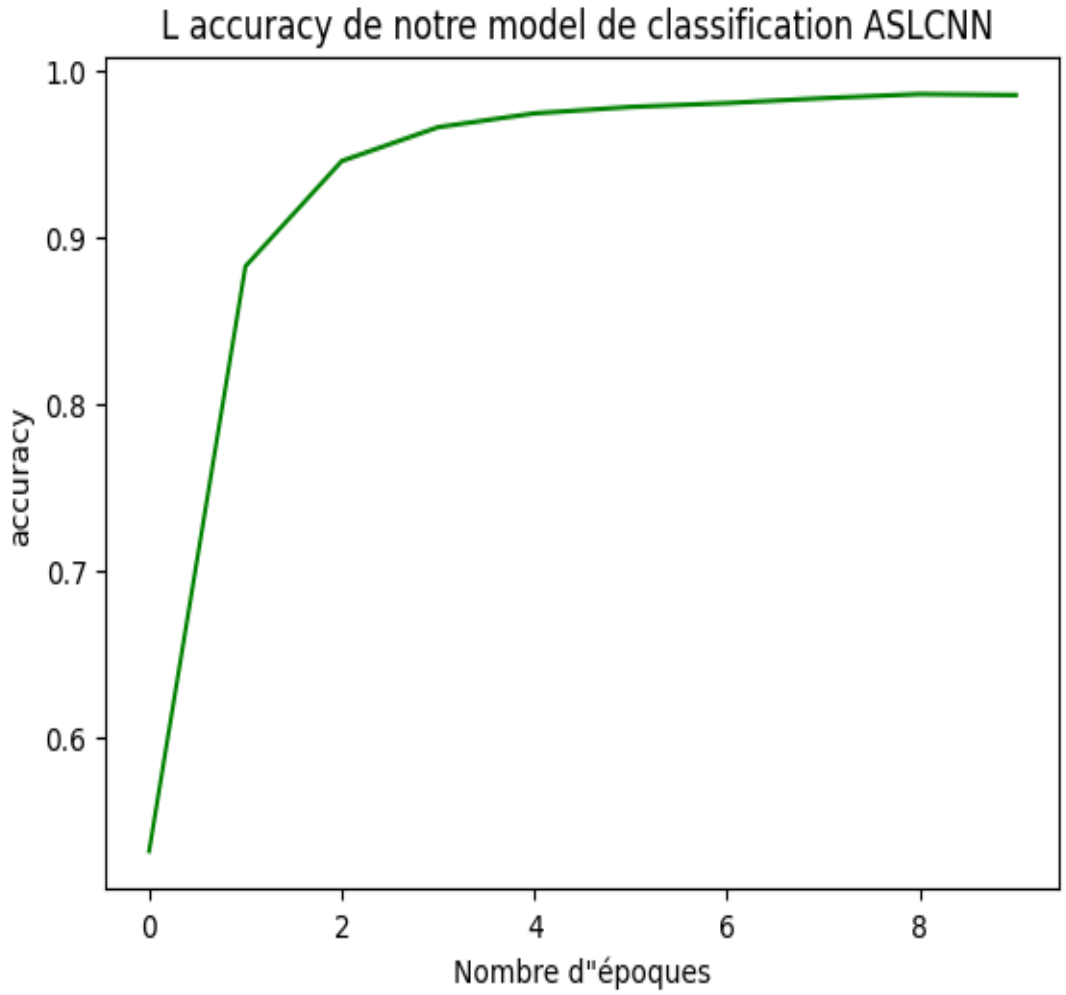


Figure4-11-l'accuracy de notre modèle 01 (ASLCNN)

9-5-2-La fonction de perte du modèle 01 (ASLCNN):

On a utilisé la fonction de perte Categorical Crossentropy : utilisée pour la classification multiclasse avec des étiquettes catégoriques qui tend vers zéro, ce qui valorise notre modèle d'annotation.

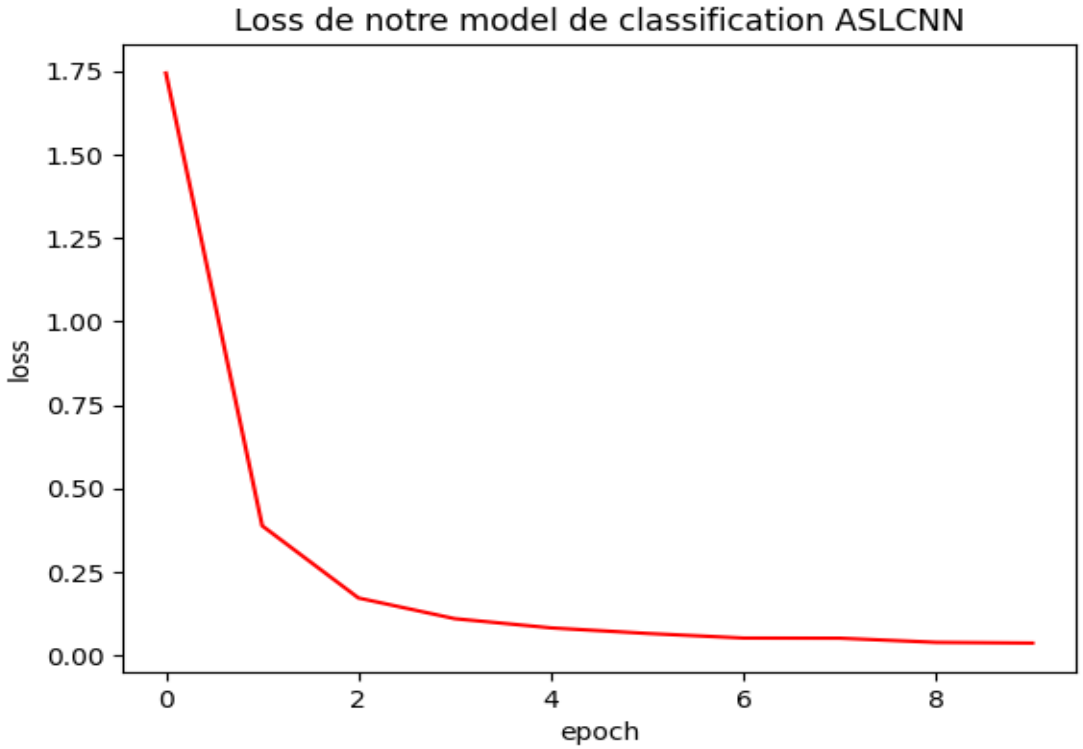


Figure4-12-la fonction loss de notre modèle 01 (ASLCNN)

Notre fonction de coût ce diminue ce que montre que le processus d'entraînement du modèle est efficace et permet une amélioration progressive de ses performances.

Notre fonction de coût qui tend vers 0 indique que le modèle 01 (ASLCNN) parvient à faire des prédictions de plus en plus précises, réduisant ainsi l'écart avec les vraies valeurs.

9-5-3- Matrice de confusion (Confusion Matrix) du modèle 01 (ASLCNN):

La dernière métrique utilisée est la matrice de confusion qui se présente comme suit :

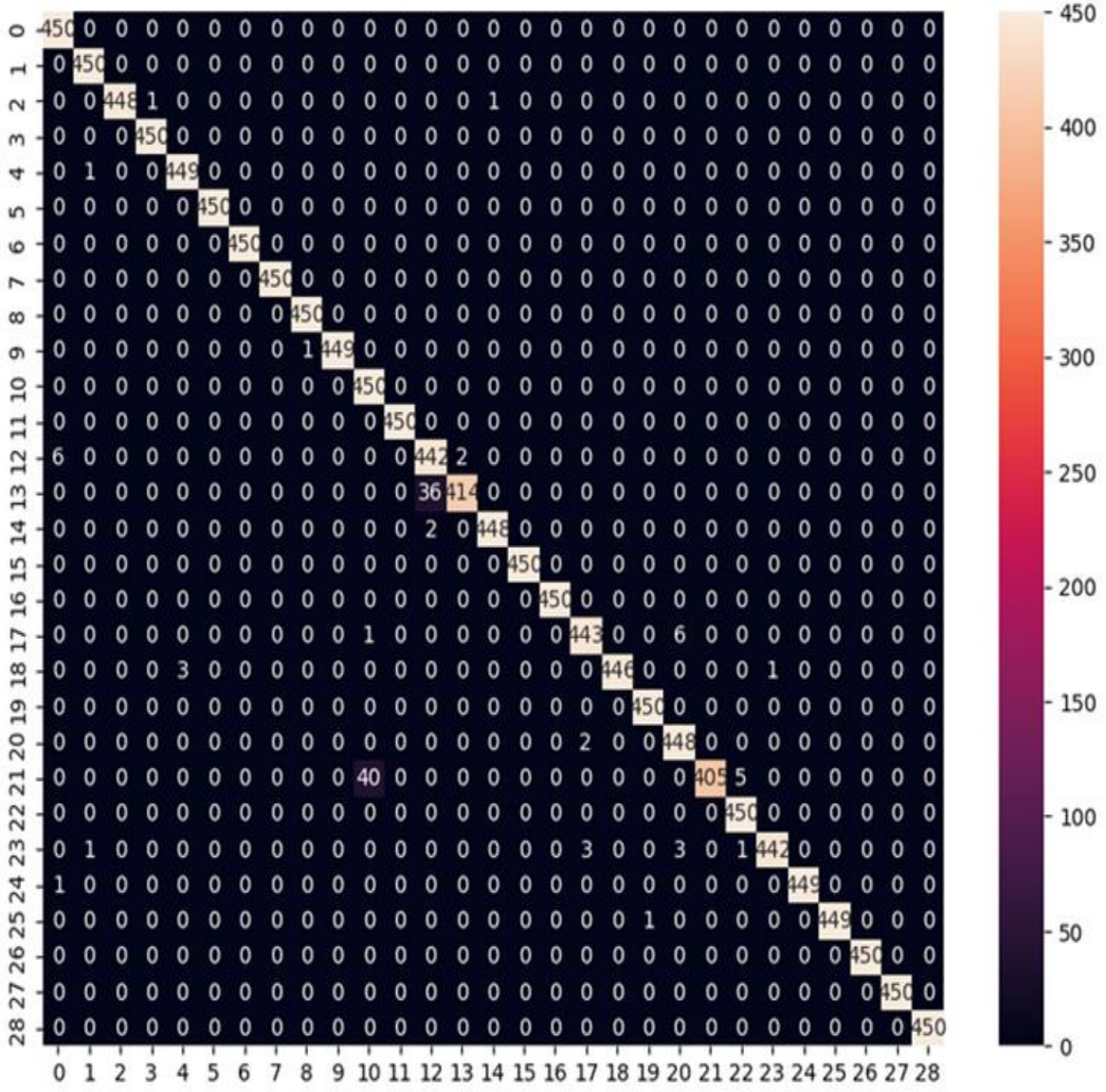


Figure4-13-la matrice de confusion de notre modèle 01 (ASLCNN)

Notre matrice de confusion présente des valeurs élevées sur la diagonale principale et des faibles valeurs hors-diagonale , cela indique effectivement une performance de classification très élevée.

10-Notre deuxième implémentation modèle 02 (RESASL) :

10-1-Codification du modèle 01 (ASLCNN) :

Nous disposons de tous les outils nécessaires, tant au niveau matériel que logiciel, pour mettre en place notre système d'annotation. Nous avons choisi d'utiliser le puissant langage de programmation Python pour cette implémentation. Voici un extrait du code utilisé pour l'implémentation du modèle 02 (RESASL)

```
#Import the libraries
import zipfile
import os
import shutil
import numpy as np
from imutils import paths

zip_ref = zipfile.ZipFile("/content/drive/MyDrive/Colab Notebooks/ĐAPM_AI/HandSign/archive3.zip", 'r')
zip_ref.extractall('/content/drive/MyDrive/resasl')
zip_ref.close()
```

Figure4-14-Une portion du code de mon modèle RESASL.

10-2-Entraînement du modèle RESASL:

le nombre d'époques est défini, nous avons essayé plusieurs choix et nous avons opté pour 5 époques (les époques sont le nombre de fois que l'apprentissage du modèle est répété),

sachant que le choix de nombre d'époque se fait généralement d'une manière stochastique.

```
Epoch 1/5
476/476 [==============================] - 665s 1s/step - loss: 0.2292 - accuracy: 0.9405 - val_loss: 69.8378 - val_accuracy: 0.0345
Epoch 2/5
476/476 [==============================] - 650s 1s/step - loss: 0.0483 - accuracy: 0.9885 - val_loss: 1.2784 - val_accuracy: 0.8243
Epoch 3/5
476/476 [==============================] - 651s 1s/step - loss: 0.0340 - accuracy: 0.9916 - val_loss: 0.0731 - val_accuracy: 0.9813
Epoch 4/5
476/476 [==============================] - 648s 1s/step - loss: 0.0202 - accuracy: 0.9949 - val_loss: 0.0410 - val_accuracy: 0.9900
Epoch 5/5
476/476 [==============================] - 650s 1s/step - loss: 0.0235 - accuracy: 0.9947 - val_loss: 0.0091 - val_accuracy: 0.9971
```

Figure4-15-Entrainement de mon modèle RESASL

10-3-Le sauvegarde de notre modèle 02 (RESASL):

On a sauvegardé notre modèle sous forme d'un fichier avec l'extension ".h5" contenant les poids et la configuration du modèle, ce qui permet de les recharger ultérieurement pour effectuer des prédictions ou poursuivre l'entraînement.

On a sauvegardé notre modèle dans un fichier nommé **RESASL.h5** par le biais de l'instruction suivante :

model.save('/content/drive/MyDrive/ColabNotebooks/ĐAPM_AI/HandSign/RESASL.h5').

l'opération inverse s'appelle le chargement du modèle elle s'effectue comme suit :

```
model=tf.keras.models.load_model("/content/drive/MyDrive/ColabNotebooks/ĐAPM_AI/HandSign/RESASL.h5")
```

10-4-Résultats expérimentaux du modèle 02 (RESASL):

Nous avons réalisé des expériences en utilisant l'optimiseur Adam pour un taux d'apprentissage (lr=0,001)et un nombre d'epoques=5 pour vérifier l'exactitude du schéma proposé, après l'entrainement la perte pour le réseau se converge vers zéro.

Nous prédisons ensuite les classes pour ces images à l'aide du modèle entraîné et estimons les performances du modèle en comparant le nombre de prédictions correctes et d'erreurs commises.

Figure4-16-Quelques résultats obtenus sur de nouvelles images inconnues.

10-5-Les métriques de mesures de performances de notre modèle (RESASL):

L'évaluation des performances de notre modèle est essentielle pour mesurer son efficacité et sa capacité à résoudre la classification. On a utilisé les trois métriques suivantes :

10-5-1-La Précision (Accuracy) du modèle RESASL:

Mon modèle a eu une précision de près de 99,71% qui est en effet un résultat extrêmement élevé et indique une performance remarquable du modèle de classification.

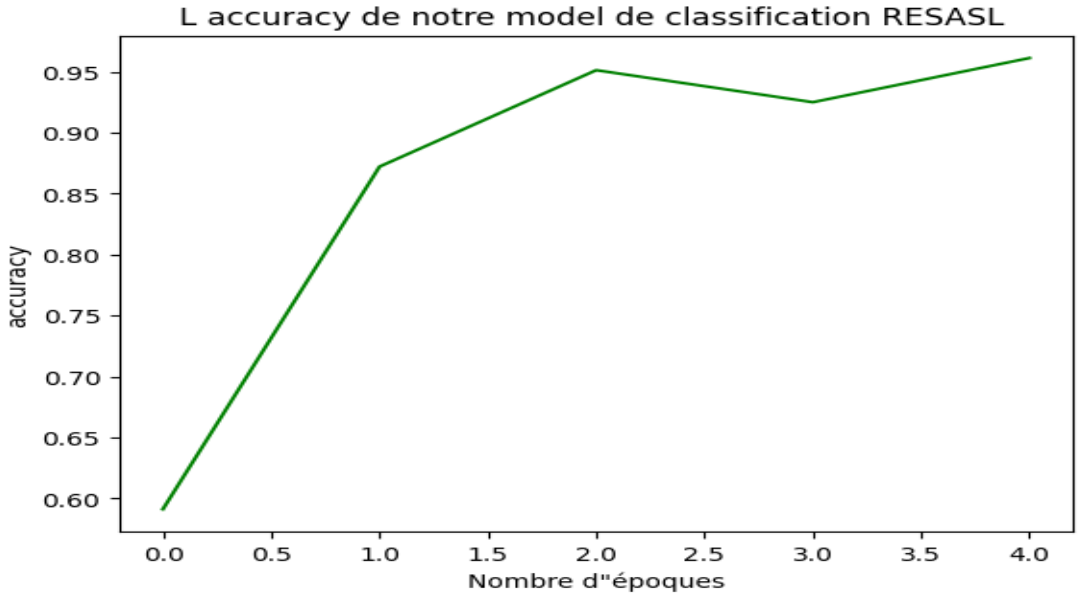


Figure4-17-l'accuracy de notre modèle RESASL

On a Une accuracy élevée, proche de 100% (99,71%), indique que le modèle a une très grande capacité à prédire la classe correcte pour la majorité des éléments. C'est un indicateur très positif de la qualité et de la robustesse du modèle RESASL.

10-5-2-La fonction de perte du modèle RESASL:

On a utilisé la fonction de perte Categorical Crossentropy : utilisée pour la classification multiclasse avec des étiquettes catégoriques qui tend vers zéro, ce qui valorise notre modèle d'annotation.

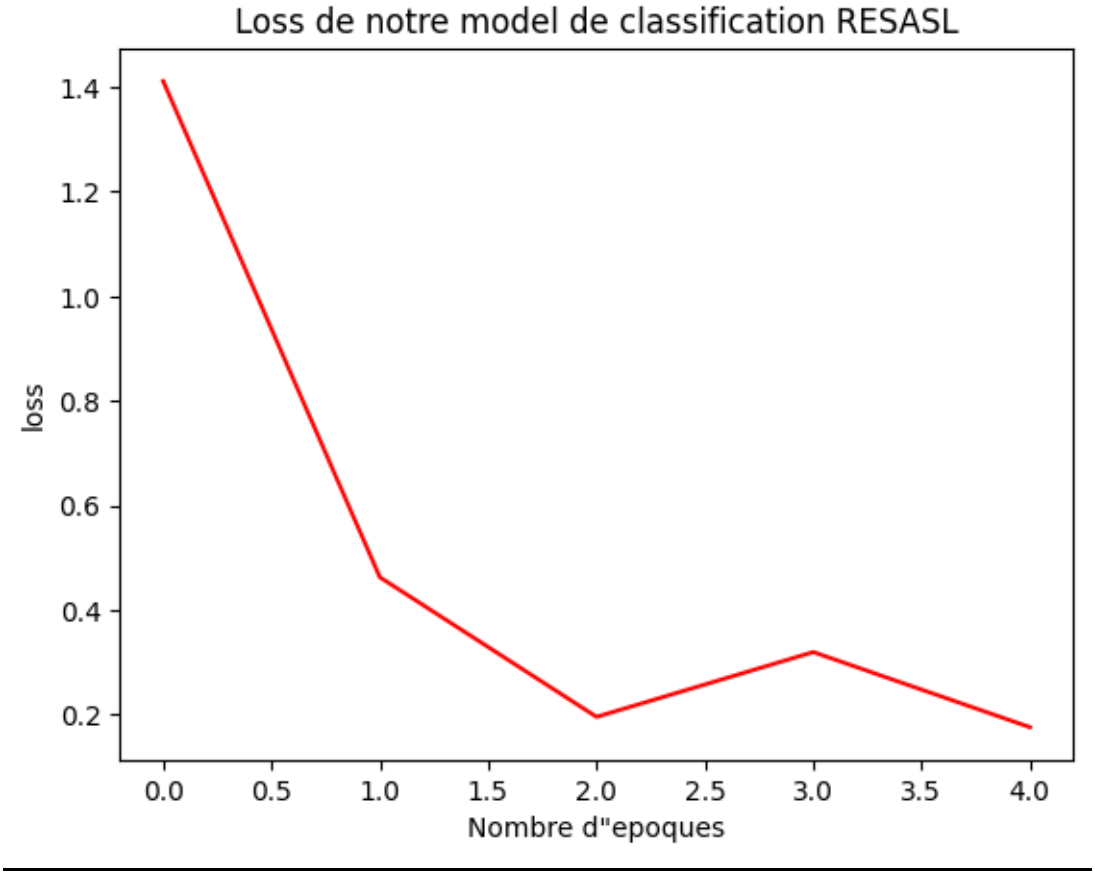


Figure4-18-la fonction loss de notre modèle RESASL.

On a une fonction de perte faible indiquant que les prédictions du modèle sont proches des vraies valeurs, et donc plus le modèle est performant (loss proche de 0 indique un modèle quasi-parfait).

10-5-3- Matrice de confusion (Confusion Matrix) du modèle RESASL:

La dernière métrique utilisée est la matrice de confusion qui se présente comme suit :

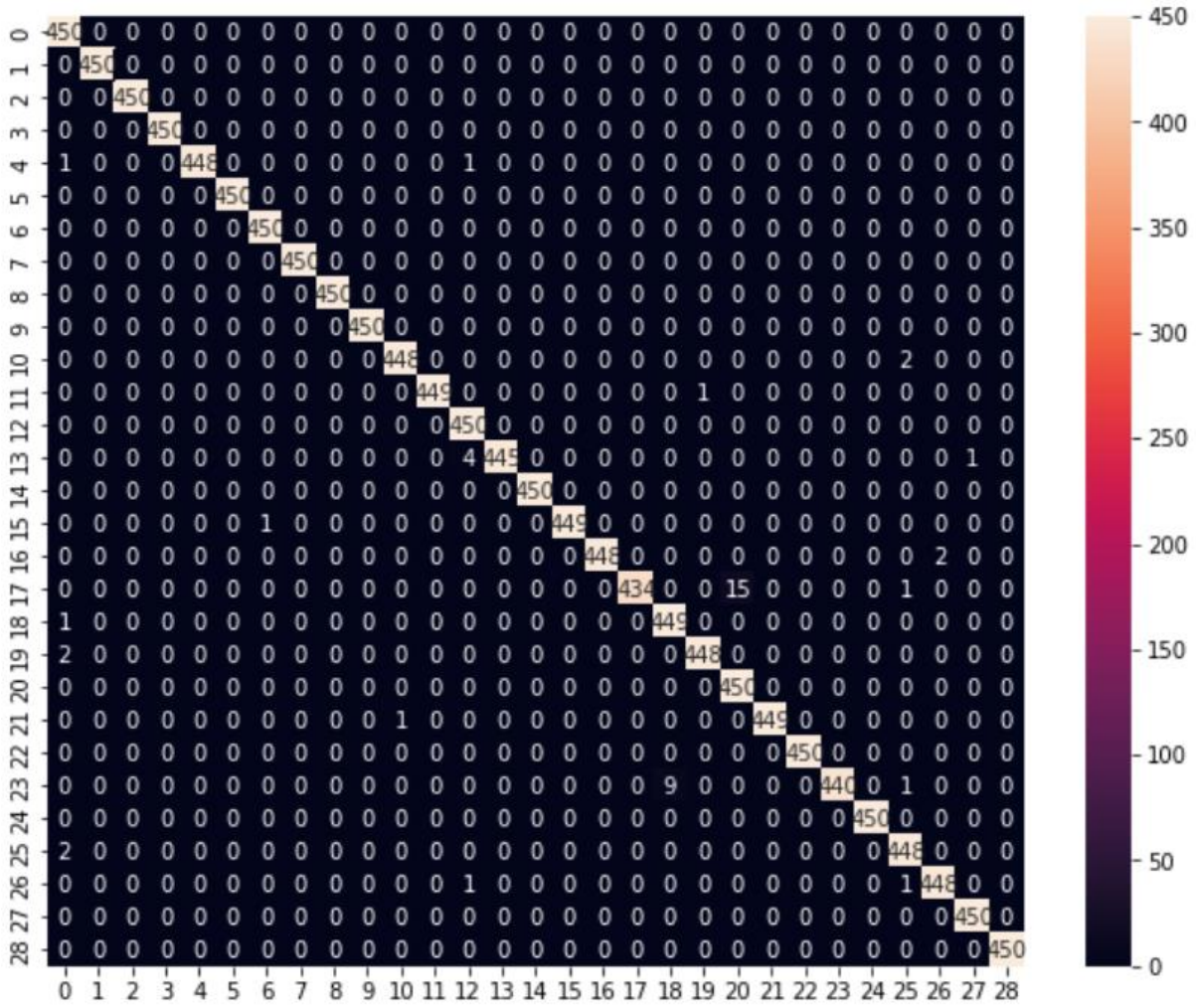


Figure4-19-la matrice de confusion de notre modèle RESASL

On remarque bien que la diagonale principale est large par rapport aux éléments hors-diagonale . Cela montre que le modèle parvient bien à classer correctement la majorité des éléments, dans les classes positive et négative et que les valeurs élevées sur la diagonale principale et les faibles valeurs hors-diagonale indiquent que le modèle est très précis et fiable dans ses prédictions.

11-Analyse et comparaison des résultats avec les travaux connexes:

Voici une comparaison des résultats des modèles ASLCNN et RESASL avec les travaux connexes sur la reconnaissance des signes de l'alphabet ASL (American Sign Language) :

Architecture	Learning Rate	Accuracy
ResNet	0.001	99.98%
EfficientNet	0.001	99.95%
RESASL	0.001	99.71%
ConvNext	0.001	99.51%
ASLCNN	0.001	99.50%
AlexNet	0.001	99.50%
VisionTransformer	0.001	88.59%
Une implémentation du CNN [12]	0.001	78,50 %

Tableau4-2-Comparaison de mes modèles avec les travaux connexes

La comparaison des résultats des modèles ASLCNN et RESASL développés dans cette étude avec les travaux connexes sur la reconnaissance des signes de l'alphabet ASL est tout simplement remarquable. Le meilleur modèle proposé jusqu'à présent, le RESNET, atteignait déjà une accuracy de 99,98% sur le même jeu de données ASL_ALPHABET. De plus, le modèle EfficientNet a ensuite repoussé les limites avec une accuracy de 99,95%, juste après, notre modèle basé sur le transfer learning, le RESASL, a atteint une accuracy de 99,71%.

Notre modèle ASLCNN conçu from scratch a presque égalé les performances des géants ConvNext et AlexNet.

Enfin, aux derniers rangs on trouve le modèle VisionTransformer affiche une accuracy de 88,59% et une autre implémentation du CNN [12] affichant une accuracy de 78.50%.

12-Conclusion :

En conclusion, le présent chapitre nous a permet de concevoir et d'implémenter deux modèles l'un from scratch (ASLCNN) et l'autre via le transfer learning (RESASL).

Les résultats obtenus sont remarquables. Les deux modèles développés ont un respectable classement par rapport aux autres études dans les tâches de classification d'images pour les systèmes de reconnaissance de la langue des signes, atteignant un taux de précision exceptionnel de (99,71% pour RESASL et 99,50% pour ASLCNN), ayant prouvé une supériorité expérimentalement et donnant des résultats convainquant et compétitifs par rapport aux méthodes profondes vues dans les travaux connexes.

Conclusion générale

Conclusion générale et perspectives :

Dans cette étude, nous avons présenté une approche utilisant des techniques d'apprentissage profond, plus précisément des réseaux neuronaux convolutifs (CNN), pour annoter efficacement les gestes représentant l'alphabet de la langue des signes américaine (ASL). Nous avons implémenté deux modèles CNN ; le premier modèle à partir de zéro (from scratch) nommé (ASLCNN) et un deuxième modèle appelé (RESASL) dont lequel on a exploré l'apprentissage par transfert (transfer learning) afin de tirer parti des capacités d'apprentissage de réseaux pré-entraînés.

Ces deux modèles présentant des performances robustes validées par des mesures d'évaluation rigoureuses atteignant un taux de précision exceptionnel de (99,71% pour RESASL et 99,50% pour ASLCNN). soulignant la fiabilité des modèles, indiquant ses potentiels pour des applications réelles visant à améliorer l'accessibilité et la communication aux personnes sourdes ou malentendantes utilisant la langue des signes Américaine.

Et en répondant à notre problématique citée dans l'introduction qui s'interroge sur la structure CNN optimale pour une prometteuse annotation j'ai trouvé plus qu'une structure et je pense qu'il est difficile de trouver la meilleure structure ou hyperparamètres de CNN, donc peut-être que la configuration de mon modèle CNN n'est pas la meilleure.

Et comme vous l'avez constaté, il n'existe pas de solution unique et universelle. La configuration optimale du modèle CNN dépend de multiples éléments, tels que la complexité des données d'entrée, la taille et la diversité de l'ensemble d'apprentissage,

les objectifs de performance visés, les ressources de calcul disponibles, etc. Il s'agit donc d'un processus itératif d'expérimentation et d'ajustement.

Tout en reconnaissant la difficulté à trouver la "meilleure" structure CNN, cette approche itérative devrait vous aider à converger vers des solutions robustes et adaptées à vos besoins spécifiques. N'hésitez pas à expérimenter, à analyser les résultats et à itérer pour affiner progressivement votre modèle..

Plutôt que de se focaliser sur une structure CNN idéale unique, il peut s'avérer plus judicieux d'adopter une démarche d'exploration et de comparaison de plusieurs architectures prometteuses. Cela permet d'identifier les configurations les plus performantes pour produire un système d'annotation très performant.

Dans nos travaux futurs, nous prévoyons de convertir nos modèles de classification d'images entraînés pour les gestes de l'alphabet de la langue des signes Américaine en un système en temps réel et d'évaluer leurs performances. Notre objectif est d'obtenir des informations sur la pertinence de ces modèles pour des applications dans le monde réel. Cela nous permettra d'identifier les défis potentiels ou les domaines à améliorer.

Et on se demande si les Réseaux Neuronaux récurrents (RNN) ou les Réseaux Antagonistes Génératifs (GAN) peuvent aider à obtenir de meilleures performances en annotation d'images.

Références

Références

Références bibliographiques

[1] Image Captioning in Deep Learning, Pranoy Radhakrishnan, 28 Sept 2017.

[2] D. Vernon, G. Metta and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents", *IEEE Trans. Evol. Comput.*, vol. 11, pp. 151-180, Apr. 2007.

[3] LeCun, Y., Bengio, Y. and Hinton, G., (2015). Deep learning. nature, 521(7553), pp.436-444.

[4] Mitchell, T.M., 1997. Machine learning. (1997). Burr Ridge, IL: McGraw Hill, 45(37), pp.870-877.

[5] CNN Based Deep Face Recognition (october 2020) Randa Aouassa PP 11

[6] Russell, S.J. and Norvig, P., (2010). Artificial Intelligence-A Modern Approach, Third International Edition.

[7] Gilles Daigle, "Système pour étiquetage automatique d'images 'mars 1999.

[8]Alsharif, B.; Altaher, A.S.; Altaher, A.; Ilyas, M.; Alalwany, E. Deep Learning Technology to Recognize American Sign Language Alphabet. *Sensors* 2023

[9] Al-Obodi, A.H.; Al-Hanine, A.M.; Al-Harbi, K.N.; Al-Dawas, M.S.; Al-Shargabi, A.A. A Saudi Sign Language recognition system based on convolutional neural networks. *Build. Serv. Eng. Res. Technol.* 2020, *13*, 3328–3334

[10] Hayani, S.; Benaddy, M.; El Meslouhi, O.; Kardouchi, M. Arab sign language recognition with convolutional neural networks. In Proceedings of the 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), Agadir, Morocco, 22–24 July 2019; IEEE: Piscataway, NJ, USA, 2019;

[11] Kasapbaşi, A.; ELBUSHRA, A.E.A.; Omar, A.H.; Yilmaz, A. DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals. *Comput. Methods Programs Biomed. Update* 2022, *2*, 100048.

[12] Mohandes, M.; Liu, J.; Deriche, M. A survey of image-based arabic sign language recognition. In Proceedings of the 2014 IEEE 11th International Multi-

Conference on Systems, Signals & Devices (SSD14), Barcelona, Spain, 11–14 February 2014;

[13] Cayamcela, M.E.M.; Lim, W. Fine-tuning a pre-trained convolutional neural network model to translate American sign language in real-time. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 100–104. [Google Scholar]

[14] Ma, Y.; Xu, T.; Kim, K. Two-Stream Mixed Convolutional Neural Network for American Sign Language Recognition. *Sensors* 2022, *22*, 5959. [Google Scholar]

[15] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham

[16] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. arXiv preprint arXiv:1705.04378, 2017.

[17] Zell, Andreas (2003). "chapter 5.2". Simulation neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley. ISBN 978-3-89319-554-1. OCLC 249017987.

[18] Prajit Ramachandran, Barret Zoph, Quoc V. Le, Searching for activation functions (2017), arXiv:1710.05941

[19] Ramachandran, P., Zoph, B. and Le, Q.V., (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

Sitographie

[w01] https://www.v7labs.com/ consulté le 08/05/2024

[w02] www.handspeak.com/learn/408/ consulté le 08/05/2024

[w03] https://datascientest.com/reinforcement-learning consulté le 30/04/2024

[w04] https://fr.isahit.com/ consulté le 08/05/2024

[w05] https://www.jedha.co/formation-ia/algorithmes-deep-learning consulté le 13/05/2024

[w06] https://www.javatpoint.com/deep-learning-algorithms consulté le 13/05/2024

[w07] https://www.simplilearn.com/image-processing-article consulté le 13/05/2024

[w08] https://larevueia.fr/quest-ce-que-la-segmentation-dimages/ consulté le 14/05/2024

[w09] https://datascientest.com/en/convolutional-neural-network-everything-you-need-to-know consulté le 14/05/2024

[w10] https://www.codecademy.com/article/deep-learning-workflow consulté le 19/05/2024

[w11] https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/ consulté le 19/05/2024

[w12] https://www.geeksforgeeks.org/what-is-the-difference-between-same-and-valid-padding-in-tf-nn-max_pool-of-tensorflow/ consulté le 26/05/2024

[w13] https://moncoachdata.com/blog/comprendre-les-reseaux-de-neurones/ consulté le 27/05/2024

[w14] https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7 consulté le 27/05/2024

[w15] https://paperswithcode.com/method/residual-block consulté le 09/06/2024

[w16] https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/ consulté le 09/06/2024

[w17] https://www.f-legrand.fr/scidoc/docmml/image/niveaux/images/images.html consulté le 10/06/2024

[w18] https://medium.com/@rndayala/gray-scale-images-8d6aacb3b761 consulté le 10/06/2024

[w19] https://www.v7labs.com/blog/image-processing-guide consulté le 10/06/2024

Figures

[f01] https://www.v7labs.com/ consulté le 08/05/2024

[f02] https://praedictia.com/page/lapprentissage-machine/quest-ce.html consulté le 13/05/2024

[f03] https://www.startasl.com/ consulté le 08/05/2024

[f04] https://www.jeveuxetredatascientist.fr/ consulté le 08/05/2024

[f05] La segmentation des images médicales en utilisant les champs de Markov cachés et la technique Deep Learning September 2020 par Melouk Anfel DOI:10.13140/RG.2.2.24745.21605

[f06] https://www.javatpoint.com/deep-learning-algorithms consulté le 13/05/2024

[f07] https://www.upgrad.com/blog/basic-cnn-architecture/ consulté le 13/05/2024

[f08] https://community.arm.com/arm-community-blogs/b/ai-and-ml-blog/posts/rnn-models-ethos-u consulté le 13/05/2024

[f09]https://datascientest.com/en/convolutional-neural-network-everything-you-need-to-know consulté le 14/05/2024

[f10] CNN Based Deep Face Recognition (october 2020) Randa Aouassa PP 11

[f11] Image Captioning in Deep Learning, Pranoy Radhakrishnan, 28 Sept 2017.

[f12] https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/

[f13] https://www.geeksforgeeks.org/what-is-the-difference-between-same-and-valid-padding-in-tf-nn-max_pool-of-tensorflow/ consulté le 26/05/2024

[f14] https://moncoachdata.com/blog/comprendre-les-reseaux-de-neurones/ consulté le 27/05/2024

[f15] https://vitalflux.com/overfitting-underfitting-concepts-interview-questions/ consulté le 29/05/2024

[f16] https://www.kaggle.com/datasets/grassknoted/asl-alphabet consulté le 08/06/2024

[f17] https://www.python.org/doc/essays/blurb/ consulté le 08/06/2024

[f18] https://www.tensorflow.org/ consulté le 08/06/2024 .

[f19] https://www.tensorflow.org/guide/keras consulté le 08/06/2024

[f20] https://paperswithcode.com/method/residual-block consulté le 09/06/2024

[f21] https://www.f-legrand.fr/scidoc/docmml/image/niveaux/images/images.html consulté le 10/06/2024

[f22]https://code-piggy.tistory.com/entry/Unity-randomColor-%EB%A7%8C%EB%93%9C%EB%8A%94-%EB%B2%95HSV-RGBA consulté le 10/06/2024

[f23] https://www.v7labs.com/blog/image-processing-guide consulté le 10/06/2024

Résumé

Résumé

Historiquement, les personnes souffrant de déficiences auditives ont été négligées, ne disposant pas des outils nécessaires pour faciliter une communication efficace. Cependant, les progrès de la technologie moderne ont ouvert la voie au développement de divers outils et logiciels visant à améliorer la qualité de vie des personnes malentendantes. Ce document de recherche présente une étude complète utilisant deux modèles distincts d'apprentissage profond pour reconnaître les gestes de la main pour l'alphabet de la langue des signes américaine (ASL).

L'objectif principal de cette étude était de tirer parti de la technologie contemporaine pour combler le fossé de la communication entre les personnes malentendantes et les personnes non malentendantes. Les modèles utilisés dans cette recherche (RESASL et ASLCNN), ont été entraînés et testés à l'aide d'un dataset comprenant plus de 87 000 images de gestes de la main pour l'alphabet ASL.

Les résultats expérimentaux de notre étude ont révélé que les deux modèles atteignant un taux de précision exceptionnel de (99,71% pour RESASL et 99,50% pour ASLCNN). soulignant ainsi leur fiabilité.

Mots-clés : Vision par ordinateur ; langue des signes américaine ; apprentissage profond ; apprentissage par transfert ; ResNet50.

الملخص

تاريخياً، تم إهمال الأشخاص الذين يعانون من ضعف السمع، حيث كانوا يفتقرون إلى الأدوات اللازمة لتسهيل التواصل الفعال. ومع ذلك، فقد مهدت التطورات في التكنولوجيا الحديثة الطريق لتطوير أدوات وبرامج مختلفة لتحسين جودة حياة الأشخاص الذين يعانون من ضعف السمع. تقدم هذه الورقة البحثية دراسة شاملة باستخدام نموذجين متميزين للتعلم العميق للتعرف على إيماءات اليدين لأبجدية لغة الإشارة الأمريكية (ASL).

كان الهدف الرئيسي من هذه الدراسة هو الاستفادة من التكنولوجيا المعاصرة لسد فجوة التواصل بين الأشخاص الذين يعانون من فقدان السمع والأشخاص الذين لا يعانون من فقدان السمع. وقد تم تدريب واختبار النموذجين المستخدمين في هذا البحث (RESASL وASLCNN) باستخدام مجموعة بيانات تحتوي على أكثر من 87000 صورة لإيماءات اليد لأبجدية لغة الإشارة الأمريكية.

وقد كشفت النتائج التجريبية لدراستنا أن كلا النموذجين حققا معدل دقة استثنائي بلغ (99.71% لـ RESASL و99.50% لـ ASLCNN)، مما يؤكد موثوقيتهما.

الكلمات المفتاحية: الرؤية الحاسوبية؛ لغة الإشارة الأمريكية؛ التعلم العميق؛ التعلم التحوّلي؛ ResNet50

Abstract

Historically, people with hearing impairments have been neglected, lacking the necessary tools to facilitate effective communication. However, advances in modern technology have paved the way for the development of various tools and software to improve the quality of life of people with hearing loss. This research paper presents a comprehensive study using two distinct deep learning models to recognise hand gestures for the American Sign Language (ASL) alphabet.

The main objective of this study was to leverage contemporary technology to bridge the communication gap between people with and without hearing loss. The models used in this research (RESASL and ASLCNN) were trained and tested using a dataset containing over 87,000 images of hand gestures for the ASL alphabet.

The experimental results of our study revealed that both models achieved an exceptional accuracy rate of (99.71% for RESASL and 99.50% for ASLCNN), underlining their reliability.

Keywords: Computer vision; American Sign Language; deep learning; transfer learning; ResNet50.