Ecole Nationale d'Ingénieurs Abderhamane Baba Touré

Département : GIT

Niveau : Master 1 Semestre1

Thème de recherche : Réseaux neuronaux (Neural Networks)

Professeur :

Abdoulaye Sidibé, PhD.

Réalisé par :

Hamady GACKOU

Abari Pascal DIARRA

Alassane FOFANA

Sommaire

Liste des Figures

Introduction Générale

Dans cette présentation, nous allons essayer de vous apprendre les bases des sujets les plus chauds dans le domaine de la science des données, c'est-à-dire l'apprentissage en profondeur. Cette présentation comprendra quatre parties. **Dans la partie 1** (grand 1), nous allons essayer de vous motiver dans ce domaine en partageant avec vous quelques applications très excitantes de l'apprentissage en profondeur. Nous allons parler brièvement des neurones dans le cerveau et comment ils inspirent les réseaux de neurones artificiels. **Dans la partie 2** (grand 2), nous continuerons à découvrir en profondeur les réseaux de neurones artificiels. **Dans la partie 3** (grand 3), nous parlerons de certaines des bibliothèques d'apprentissage en profondeur, à savoir **Keras**, **PyTorch**, et **TensorFlow**. Et nous apprendrons comment utiliser la bibliothèque **Keras** pour construire des modèles de prédiction. **Dans la partie 4** (grand 4), nous apprendrons les réseaux de neurones profonds supervisés et non supervisés, notamment les réseaux de neurones convolutifs, les réseaux de neurones récurrents et les auto-encodeurs. Nous apprendrons également à utiliser la bibliothèque **Keras** pour construire un réseau de neurones convolutifs. Nous terminerons par un projet final **(partie 5),** qui consistera à produire un modèle de deep learning pour un problème concret basé sur la bibliothèque **Keras.**

Maintenant bienvenue et commençons !

I. Introduction à l'apprentissage profond

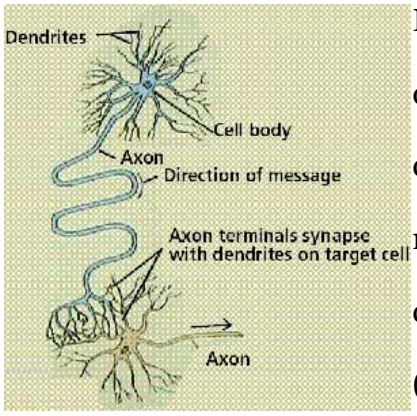
(Introduction to Deep Learning)

1. Introduction à l'apprentissage profond
(Introduction to Deep Learning)

Nous allons commencer à parler du deep learning et comment les récentes avancées dans le domaine ont conduit à des applications étonnantes et époustouflantes. Nous sommes sûr que vous êtes conscient que l'apprentissage en profondeur est l'un des sujets les plus chauds en science des données, sinon le plus chaud, surtout avec la quantité énorme de projets fascinants qui font surface avec l'aide de l'apprentissage profond; projets que les gens ont jugé presque impossible avec un peu plus d'une décennie. Par conséquent, il y a beaucoup d'excitation à propos de l'apprentissage profond. Nous allons partager avec vous quelques applications étonnantes et récentes de deep learning qui vous inspireront et motiveront même plus sur le deep learning. La première application étonnante est la restauration des couleurs, où une image donnée en niveaux de gris est automatiquement transformée en une image colorée. Une autre application vraiment cool mais double épée est la mise en œuvre de discours, où un clip audio est synthétisé avec une vidéo, et les mouvements de lèvre dans la vidéo sont synchronisés avec les sons et les mots dans le clip audio. Une autre application fascinante du deep learning est la génération automatique de l'écriture manuscrite.

2. Réseaux neuronaux biologiques

(Biological Neural Networks)



Le cerveau est principalement composé d'environ **10 milliards de neurones**, chacun connecté à environ 10 000 autres neurones. Chacune des gouttes jaunes dans l'image ci-dessus sont des corps cellulaires neuronaux (soma), et les lignes sont les canaux d'entrée et de sortie (dendrites et axones) qui les relient. Chaque neurone reçoit **des entrées électrochimiques** d'autres neurones au niveau des dendrites. Si la somme de ces entrées électriques est suffisamment puissante pour activer le neurone, il transmet un signal électrochimique le long de l'axone, et transmet ce signal aux autres neurones dont les dendrites sont attachées à l'une des terminaisons axonales. Ces neurones attachés peuvent alors se déclencher. Il est important de noter qu'un **neurone ne se déclenche que si le signal total reçu au niveau du corps cellulaire dépasse un certain niveau.** Le neurone se déclenche ou ne se déclenche pas, il n'y a pas différents niveaux de déclenchement. Ainsi, tout notre cerveau est composé de ces neurones de transmission électrochimique interconnectés. À partir d'un très grand nombre d'unités de traitement extrêmement simples (chacune effectuant une somme pondérée de ses entrées, puis émettant un signal binaire si l'entrée totale dépasse un certain niveau), le cerveau parvient à effectuer des tâches extrêmement complexes. C'est le modèle sur lequel reposent les réseaux de neurones artificiels. Jusqu'à présent, les réseaux de neurones artificiels n'ont même pas réussi à modéliser la complexité du cerveau, mais ils se sont avérés

bons pour des problèmes faciles pour un humain mais difficiles pour un ordinateur traditionnel.

3. Introduction aux Réseaux neuronaux artificiels
(Introduction to Artificial Neural Networks)

Les réseaux de neurones artificiels (ANN) sont des algorithmes basés sur la fonction cérébrale et sont utilisés pour modéliser des modèles complexes et prévoir des problèmes. Le réseau de neurones artificiels (ANN) est une méthode d'apprentissage en profondeur issue du concept de réseaux de neurones biologiques du cerveau humain. Le développement de l'ANN est le résultat d'une tentative de reproduire le fonctionnement du cerveau humain. Le fonctionnement des ANN est extrêmement similaire à celui des réseaux de neurones biologiques, bien qu'ils ne soient pas identiques. L'algorithme ANN n'accepte que des données numériques et structurées. Les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récursifs (RNN) sont utilisés pour accepter des formes de données non structurées et non numériques telles que l'image, le texte et la parole.

II. Réseaux neuronaux artificiels

(Artificial Neural Networks**)**

1. Architecture des réseaux de neurones artificiels

(Artificial Neural Networks Architecture)

Il y a trois couches dans l'architecture du réseau : la couche d'entrée, la couche cachée (plusieurs) et la couche de sortie. En raison des nombreuses couches, on parle parfois de MLP (Multi-Layer Perceptron).

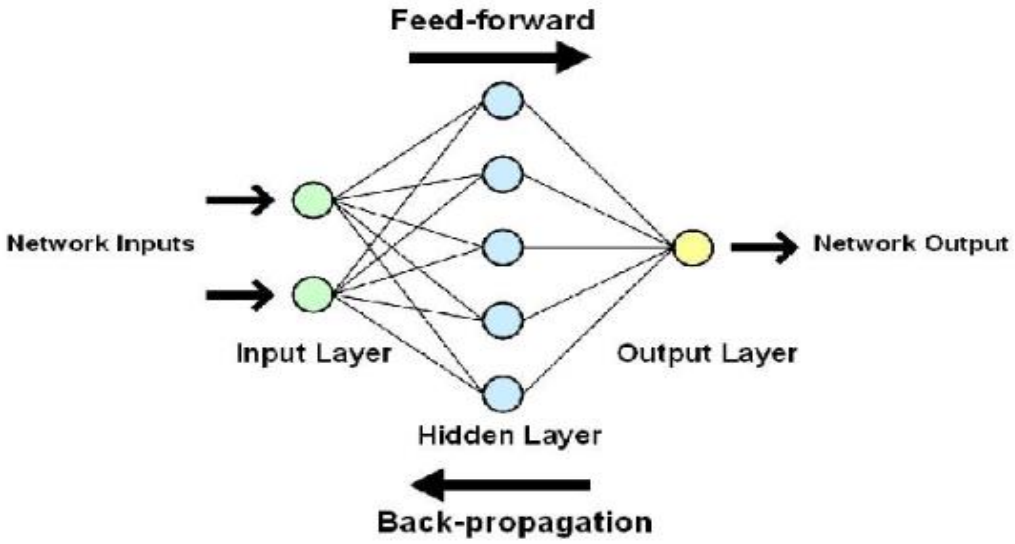


Fig1. : Architecture simplifié d'un réseau artificiel

Il est possible de considérer la couche cachée comme une « couche de distillation », qui extrait certains des modèles les plus pertinents des entrées et les envoie à la couche suivante pour une analyse plus approfondie. Il accélère et améliore l'efficacité du réseau en reconnaissant uniquement les informations les plus importantes des entrées et en supprimant les informations redondantes.

2. Algorithme de descente de gradient
(Gradient Descent)

L'algorithme de descente de gradient calcule de manière itérative le point suivant en utilisant le gradient à la position actuelle, le met à l'échelle (par un taux d'apprentissage) et soustrait la valeur obtenue de la position actuelle (fait un pas). Il soustrait la valeur car nous voulons minimiser la fonction (la maximiser serait additionner). Ce processus peut s'écrire : $p_{n+1} = p_n - \eta \nabla f(p_n)$. Il existe un paramètre important **η** qui redimensionne le gradient et contrôle ainsi la taille du pas. En machine learning, on l'appelle **taux d'apprentissage** et a une forte influence sur les performances. Plus le taux d'apprentissage est petit, plus le **GD** converge ou peut atteindre une itération maximale avant d'atteindre le point optimal. Si le taux d'apprentissage est trop élevé, l'algorithme peut ne pas converger vers le point optimal (sauter) ou même diverger complètement. En résumé, les étapes de la méthode Gradient Descent sont :

1. choisir un point de départ (initialisation)
2. calculer le gradient à ce point
3. faire un pas gradué dans le sens opposé au dégradé (objectif : minimiser)
4. répétez les points 2 et 3 jusqu'à ce que l'un des critères soit rempli :

- nombre maximum d'itérations atteint
- la taille du pas est inférieure à la tolérance (en raison de la mise à l'échelle ou d'un petit gradient). Voici le lien ci-dessous d'un code expliqué qui implémente l'algorithme de la descente de gradient **fromScratch :** https://github.com/gackouhamady/DescentDeGradientFromScratch/blob/main/GradientDescent.ipynb

3. Propagation vers l'avant
(Forward Propagation)

En termes de réseau de neurones, la propagation vers l'avant est importante et elle aidera à décider si les poids attribués sont bons à apprendre pour l'énoncé du problème donné. Techniquement, deux étapes majeures sont effectuées dans la propagation vers l'avant :

- Additionner le produit : cela signifie multiplier le vecteur de poids avec le vecteur d'entrée donné. Et puis ça continue jusqu'à la dernière couche où nous prenons la décision.
- Passer la somme par la fonction d'activation : la somme du produit du poids et du vecteur d'entrée est passée dans chaque couche pour vous donner la couche de sortie. Et, ensuite, la sortie d'une couche devient l'entrée de la couche suivante à multiplier par le vecteur de poids dans cette couche. Et ce processus se poursuit jusqu'à la fonction d'activation de la couche de sortie.

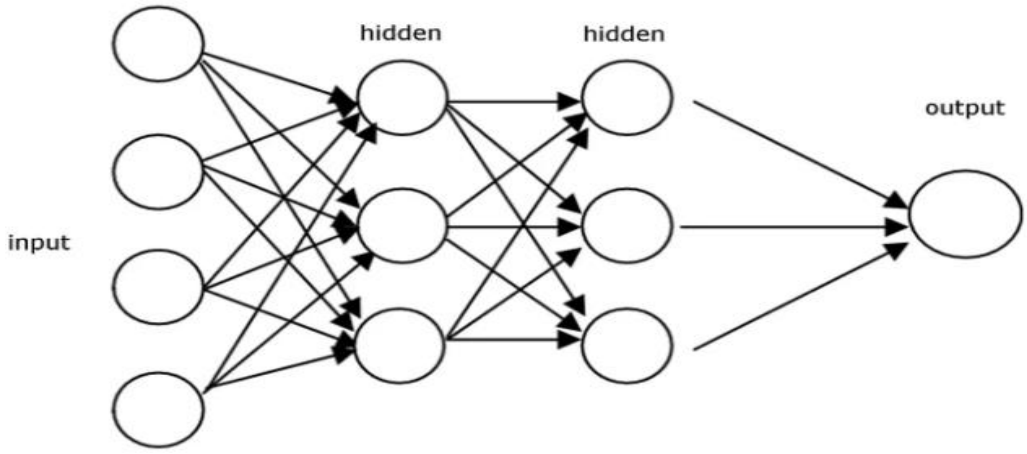


Fig2. : Réseau neuronal d'anticipation

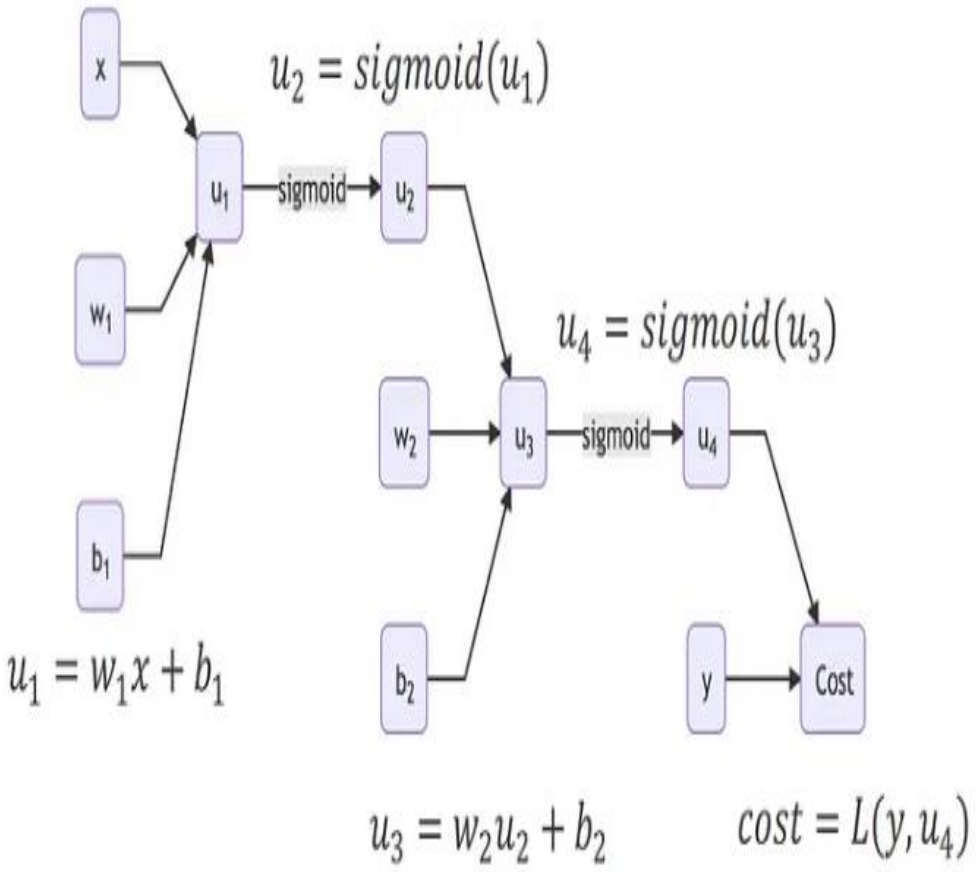


Fig3 : Forward Propagation

Dans ce réseau donné, l'anticipation est facile à comprendre. x est l'entrée initiale, b1 est le biais et w1 est le poids. Chaque couche a sigmoïde comme fonction d'activation. La valeur prédite est u4 et le coût est l'erreur/perte calculée. La somme de la production est u1, u3, u2 et u4 sont émis après l'application de la fonction d'activation, c'est-à-dire sigmoïde dans ce cas. Nous espérons que cela aidera à comprendre le réseau d'anticipation jusqu'au calcul de la perte.

4. Rétropropagation
(Backpropagation)

Dans l'apprentissage automatique, la propagation vers l'arrière est l'un des algorithmes importants pour la formation du réseau d'alimentation en aval. Une fois que nous avons traversé le réseau de transfert, nous obtenons une sortie prédite à comparer avec la sortie cible. Sur cette base, nous avons compris que nous pouvions calculer la perte totale et dire si le modèle était bon ou non. Sinon, nous utilisons la valeur de perte pour recalculer à nouveau les poids pour la passe avant. De plus, ce processus de recalcul du poids est rendu simple et efficace grâce à la rétropropagation. La rétropropagation calcule le gradient de la fonction de perte par rapport aux poids du réseau.

Voici comment procéder :

- Calculez la différence d'erreurs entre la sortie attendue et la sortie prédite reçue en passe avant. Ce processus continuera à donner l'Erreur/Perte à chaque passage vers l'avant. Si la perte/l'erreur est acceptable, vous pouvez stocker le modèle à tester avec des données de test.
- Obtenir le delta : multipliez l'erreur par la dérivée pour obtenir le delta
- Somme du produit : multipliez le delta par le vecteur d'entrée et ainsi de suite pour obtenir la somme du produit.

Termes à comprendre pour la propagation vers l'avant et vers l'arrière : produit scalaire, dérivée, règle de chaîne, gradient, calcul de perte/erreur et sa méthode.

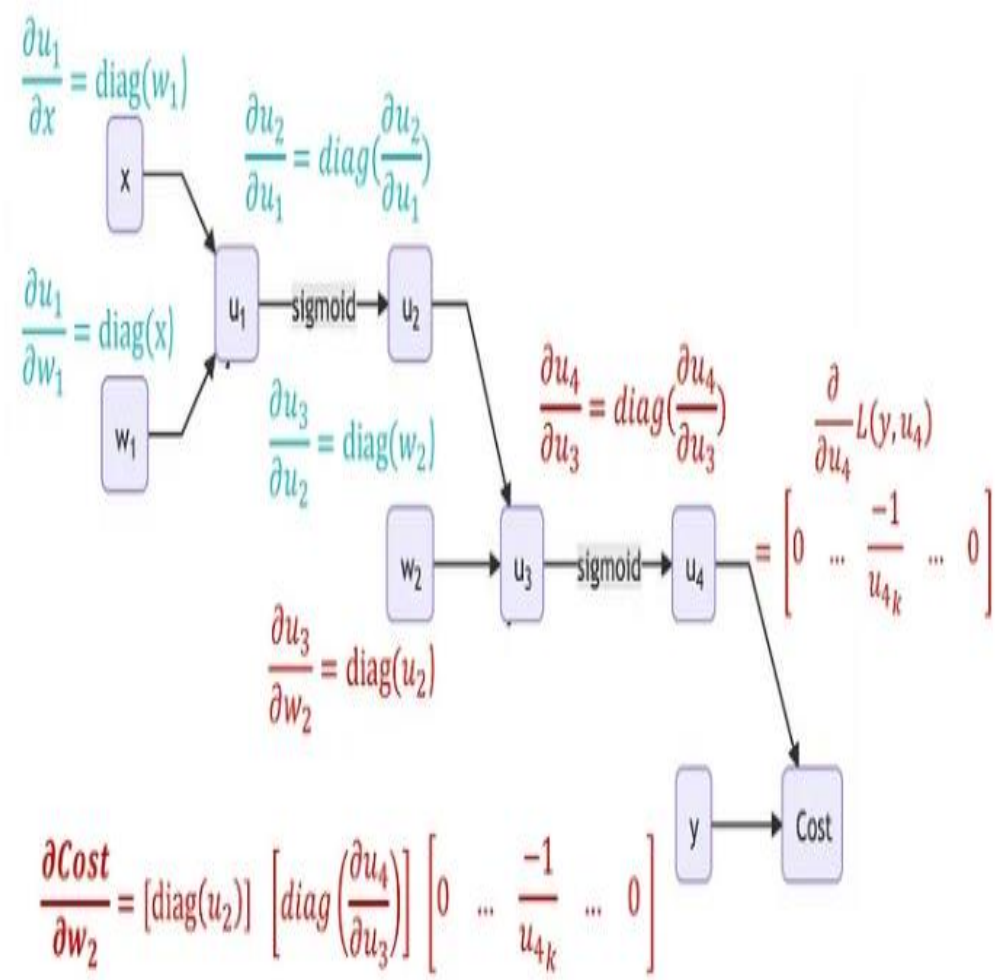


Fig4. : Back Ward Propagation

Après le passage vers l'avant et le calcul des coûts (erreur/perte), la rétropropagation a lieu pour obtenir le delta. Pour calculer le delta, il applique une dérivée partielle et il part de la sortie jusqu'à la dernière couche. Et, une fois le delta calculé pour la dernière couche, la formule ci-dessous est appliquée pour obtenir le nouveau poids.

Nouveau poids = Ancien poids - delta * taux d'apprentissage.

Ici, le taux d'apprentissage peut être n'importe quelle valeur supposée entre 0 et 1. Une fois que le nouveau poids est calculé pour la dernière couche, ce processus passe à la couche cachée précédente pour calculer le delta et le nouveau

poids. Cela utilise la règle de la chaîne pour obtenir le delta de la couche cachée, comme nous pouvons le voir sur l'image jointe ci-dessous. La formule en rouge montre la formule directe via la règle de chaîne et la dérivée partielle.

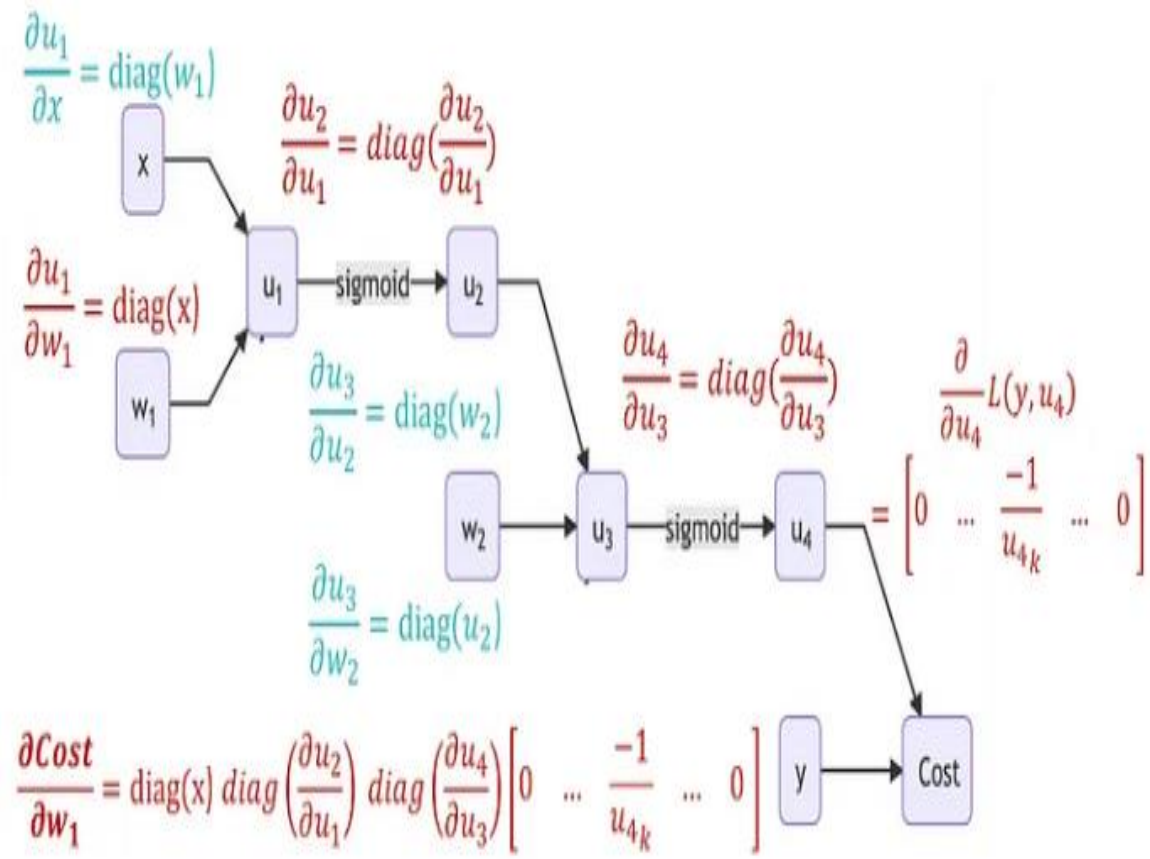


Fig.5 : Règle de la Chaine Directe

5. Fonctions d'activations
(Activations Functions)

Dans le domaine des réseaux de neurones artificiels, la **fonction d'activation** est une fonction mathématique appliquée à un signal en sortie d'un neurone artificiel. Le terme de "fonction d'activation" vient de l'équivalent biologique "potentiel d'activation", seuil de stimulation qui, une fois atteint entraîne une réponse du neurone (source Wikipédia).

6. Application des réseaux de neurones artificiels
(Application of Artificial Neural Networks)

Les ANN ont un large éventail d'applications en raison de leurs propriétés uniques. Quelques-unes des applications importantes des ANN comprennent :

- **Traitement d'image et reconnaissance de caractères :**

Les ANN jouent un rôle important dans la reconnaissance des images et des caractères en raison de leur capacité à accepter de nombreuses entrées, à les traiter et à déduire des corrélations non linéaires cachées et compliquées. La reconnaissance de caractères, telle que la reconnaissance de l'écriture manuscrite, a de nombreuses applications dans la détection de la fraude (par exemple, la fraude bancaire) et même dans les évaluations de la sécurité nationale.

Fig6. : Image Processing and Character recognition

La reconnaissance d'images est une discipline en évolution rapide avec plusieurs applications allant de l'identification faciale sur les réseaux sociaux à la détection du cancer en médecine, en passant par le traitement d'images satellite à des fins agricoles et de défense. Les réseaux de neurones profonds, qui forment le cœur de «l'apprentissage en profondeur», ont maintenant ouvert toutes les avancées nouvelles et transformatrices en vision par ordinateur, reconnaissance de la parole et traitement du langage naturel - des exemples notables étant les véhicules autonomes, grâce à la recherche ANN.

- **Prevision :**

Les prévisions sont largement utilisées dans les décisions quotidiennes des entreprises (ventes, répartition financière entre les biens et utilisation des capacités), la politique économique et monétaire, la finance et le marché boursier. Les problèmes de prévision sont souvent complexes; par exemple, la prévision des prix des actions est compliquée avec de nombreuses variables sous-jacentes (certaines connues, d'autres invisibles).

Fig7. : Forecasting

7. Avantages des réseaux de neurones artificiels

(Advantages of Artificial Neural Networks)

- Les paires attribut-valeur sont utilisées pour représenter les problèmes dans ANN.
- La sortie des ANN peut être à valeurs discrètes, à valeurs réelles ou un vecteur de plusieurs caractéristiques à valeurs réelles ou discrètes, tandis que la fonction cible peut être à valeurs discrètes, à valeurs réelles ou un vecteur de nombreuses valeurs réelles ou discrètes.
- Le bruit dans les données d'apprentissage n'est pas un problème pour les techniques d'apprentissage ANN. Il peut y avoir des erreurs dans les échantillons d'apprentissage, mais elles n'affecteront pas le résultat final.
- Il est utilisé lorsqu'une évaluation rapide de la fonction cible enseignée est nécessaire.
- Le nombre de pondérations dans le réseau, le nombre d'instances de formation évaluées et les paramètres des différents paramètres d'algorithme d'apprentissage peuvent tous contribuer à prolonger les périodes de formation des ANN.

8. Inconvénients des réseaux de neurones artificiels

(Disadvantages of Artificial Neural Networks)

- **Dépendance matérielle :** La construction de réseaux de neurones artificiels nécessite l'utilisation de processeurs parallèles. De ce fait, la réalisation de l'équipement est contingente.

- **Comprendre le fonctionnement du réseau :** C'est le problème le plus sérieux avec ANN. Lorsqu'ANN fournit une réponse approfondie, il n'explique pas pourquoi ni comment il a été choisi. En conséquence, la confiance du réseau est érodée.
- **Structure de réseau assurée :** Aucune règle précise ne détermine la structure des réseaux de neurones artificiels. L'expérience et les essais et erreurs sont utilisés pour développer une structure de réseau appropriée.

- **Difficulté à présenter le problème au réseau :** Les ANN sont capables de travailler avec des données numériques. Avant d'être introduits à ANN, les problèmes doivent être convertis en valeurs numériques. La méthode d'affichage choisie aura un impact direct sur les performances du réseau. La compétence de l'utilisateur est un facteur ici.
- **La durée de vie du réseau est inconnue :** Lorsque l'erreur du réseau sur l'échantillon est réduite à un montant spécifique, la formation est terminée. La valeur ne produit pas les meilleurs résultats.

9. Application pratique :

- **Lien du code expliqué :** Cliquer sur le lien ci-dessous en accédant au notebook du code expliquant l'ANN (Forward Propagation) dans github et l'ouvrir dans **colab**:

  https://github.com/gackouhamady/ANN-ArtificialNeuralNetworks/blob/main/ArtificialNeuralNetworks.ipynb

- **Lien de la pratique :** Cliquer sur le lien ci-dessous en accédant au notebook du code pratique sur ANN (Forward Propagation) dans github et l'ouvrir dans **colab**:

  https://github.com/gackouhamady/ForwardPropagationPratice/blob/main/Forward_Propgation_py_v1_0.ipynb

10 . Conclusion:

Les réseaux de neurones analytiques (ANN) sont des modèles puissants qui peuvent être appliqués dans de nombreux scénarios. Plusieurs utilisations notables des RNA ont été mentionnées ci-dessus, bien qu'elles aient des applications dans diverses industries, notamment la médecine, la sécurité/la finance, le gouvernement, l'agriculture et la défense.

III. Bibliothèques d'apprentissage en profondeur

(Deep Learning Librairies)

1. Les Bibliothèques fondamentales

Avant que nous puissions commencer à construire en profondeur des réseaux d'apprentissage, nous passerons quelques temps d'apprendre les différentes bibliothèques d'apprentissage en profondeurs. Nous allons couvrir brièvement trois bibliothèques fondamentales en deep learning. Ces bibliothèques sont **TensorFlow**, **Ke**ras et **PyTorch**.

TensorFlow est le plus populaire. C'est la bibliothèque qui est principalement utilisée dans la production de profondes modèles d'apprentissage. Il a une très grande communauté. Juste un rapide coup d'œil sur nombre de **forks sur le Github** de la librairie référentiel, ainsi le nombre des **commits** et **pull requests** devraient suffire pour vous donner une idée de la popularité de la bibliothèque. Il a été développé par Google et rendu public en 2015. **PyTorch**, d'autre part, est le cousin du Framework **Torch**, prend en charge des algorithmes s'exécutant sur les **GPU** en particulier. **PyTorch** est sorti en 2016. **PyTorch** et **TensorFlow** ne sont pas faciles à utiliser. Donc pour les personnes qui commencent tout juste à apprendre le deep learning, il n'y a pas mieux de bibliothèque à utiliser autre que **Keras** bibliothèque. **Keras** est une API de haut niveau pour la construction de modèles d'apprentissage en profondeur. Il a gagné la faveur pour sa facilité d'utilisation. La construction d'un réseau d'apprentissage

très complexe peut être réalisée avec **Keras** avec seulement quelques lignes de code.

Keras fonctionne normalement au-dessus d'une bibliothèque de bas niveau telle que TensorFlow. Cela signifie que pour pouvoir utiliser le Bibliothèque **Keras**, vous devrez installer **TensorFlow**. **Keras** est également pris en charge par Google (créée par **François Chollet**).

2. La Bibliothèque Keras
(Keras Library)

Keras est une API d'apprentissage en profondeur écrite en Python, s'exécutant sur la plate-forme d'apprentissage automatique TensorFlow . Il a été développé dans le but de permettre une expérimentation rapide. Pouvoir passer de l'idée au résultat le plus rapidement possible.

Keras est :

- Simple -- mais pas simpliste. Keras réduit la charge cognitive du développeur pour vous permettre de vous concentrer sur les parties du problème qui comptent vraiment.
- Puissant - Keras offre des performances et une évolutivité de pointe : il est utilisé par des organisations et des entreprises telles que la NASA, YouTube ou Waymo.

Pour plus de détails, voici le lien vers l'API officiel **Keras** :

https://keras.io/api/

IV. Modèles d'apprentissage en profondeur

(Deep Learning models)

Dans cette section, nous découvrirons la différence entre les réseaux de neurones superficiels et profonds. Nous découvrirons également les réseaux convolutifs et comment les construire à l'aide de la bibliothèque Keras. Enfin, nous découvrirons également les réseaux de neurones récurrents et les auto-encodeurs

1. Réseaux de neurones peu profonds ou profonds (Shallow Versus Deep Neural Networks)

Les réseaux de neurones peu profonds servent de pierre de construction de **profonds réseaux de neurones** et sont plus faciles à comprendre en raison de leur simplicité. Un réseau de neurones avec 1 ou 2 couche(s) cachée(s) est considéré comme **un réseau de neurones peu profond** alors qu'un réseau avec de nombreuses couches cachées (>2) est considéré comme un **réseau neuronal profond**. Aussi, contrairement à **un réseau de neurones peu profond** qui ne prend qu'en entrée de vecteurs numériques, les **réseaux de neurones profonds** sont capables de prendre des données brutes telles que des images et le texte et extraire automatiquement les fonctionnalités nécessaires pour apprendre mieux les données .**Les réseaux de neurones profond** fonctionnent mieux lorsqu'ils sont entraînés avec de grandes quantités de données, de grandes quantités de données doivent être utilisées pour éviter le **sur ajustement** des données d'entraînement. Maintenant que de grandes quantités de données sont facilement disponible et facile à acquérir comme jamais auparavant, les algorithmes d'apprentissage en profondeur sont

essayés et testés comme jamais auparavant. Enfin, cela va de pair avec les machines à processeurs GPU.

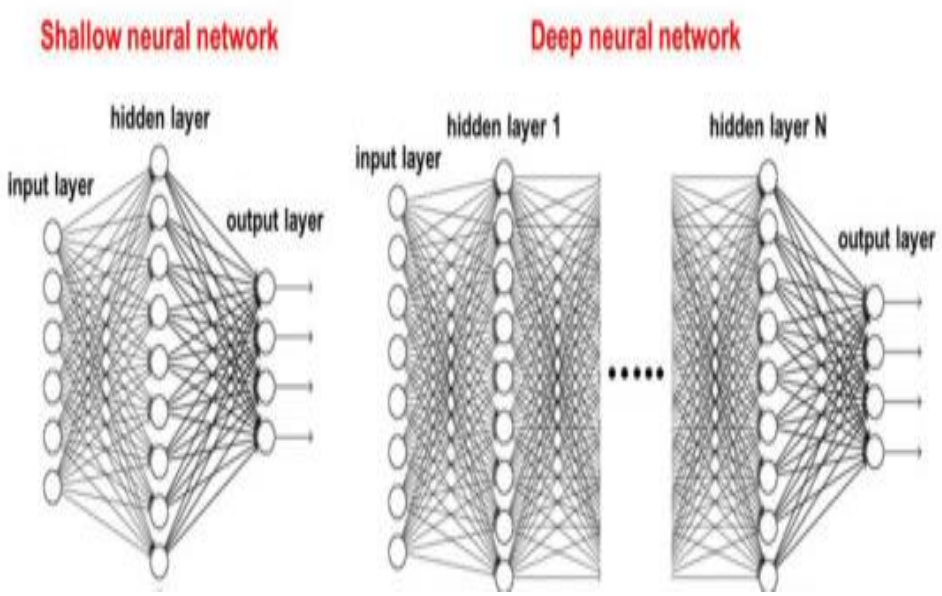


Fig.8: Shallow Versus Deep Neural Networks

2. Réseaux de neurones convolutifs (Convolutional Neural Networks)

- **Contexte des CNN**

Les **CNN** ont été développés et utilisés pour la première fois dans les années 1980. Le plus qu'un **CNN** pouvait faire à cette époque était de reconnaître les chiffres manuscrits. Il était principalement utilisé dans les secteurs postaux pour lire les codes postaux, les codes PIN, etc. La chose importante à retenir à propos de tout modèle d'apprentissage en profondeur est qu'il nécessite une grande quantité de données pour s'entraîner et nécessite également beaucoup de ressources informatiques. C'était un inconvénient majeur pour les **CNN** à cette époque et, par conséquent, les **CNN** n'étaient limités qu'aux secteurs postaux et n'avaient pas réussi à entrer dans le monde de l'apprentissage automatique. En 2012, **Alex**

Krizhevsky s'est rendu compte qu'il était temps de ramener la branche de l'apprentissage en profondeur qui utilise des réseaux de neurones multicouches. La disponibilité de grands ensembles de données, pour être plus spécifiques des ensembles de données ImageNet avec des millions d'images étiquetées et une abondance de ressources informatiques, a permis aux chercheurs de faire revivre les **CNN.**

- **Qu'est-ce qu'un CNN exactement?**

Dans l'apprentissage en profondeur, un **réseau de neurones convolutifs** (**CNN/ConvNet**) est une classe de réseaux de neurones profonds, le plus souvent appliqués pour analyser l'imagerie visuelle. Maintenant, quand nous pensons à un réseau de neurones, nous pensons aux multiplications matricielles, mais ce n'est pas le cas avec **ConvNet**. Il utilise une technique spéciale appelée Convolution. Or, en mathématiques, **la convolution** est une opération mathématique sur deux fonctions qui produit une troisième fonction qui exprime comment la forme de l'une est modifiée par l'autre.

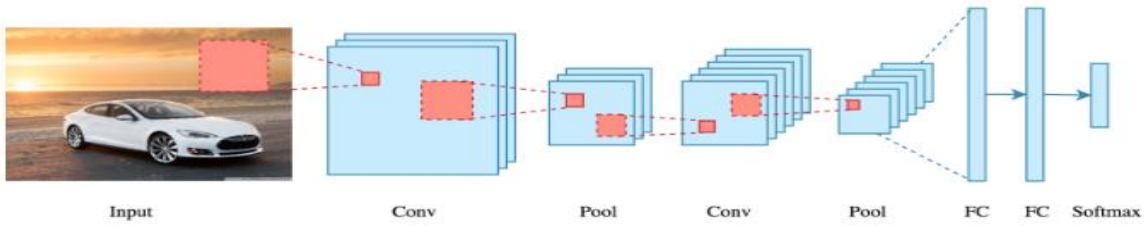


Fig.9: CNN Illustration

Mais nous n'avons pas vraiment besoin d'aller au-delà de la partie mathématique pour comprendre ce qu'est un CNN ou comment il fonctionne. *L'essentiel est que le rôle du ConvNet est de réduire les images sous une forme plus facile à traiter, sans perdre les fonctionnalités essentielles pour obtenir une bonne prédiction.*

- **Comment CNN marche ?**

Avant d'aborder le fonctionnement de CNN, couvrons les bases telles que ce qu'est une image et comment est-elle représentée. Une image RVB n'est rien d'autre qu'une matrice de valeurs de pixels ayant trois plans alors qu'une image en niveaux de gris est la même mais elle a un seul plan. Jetez un œil à cette image pour en savoir plus.

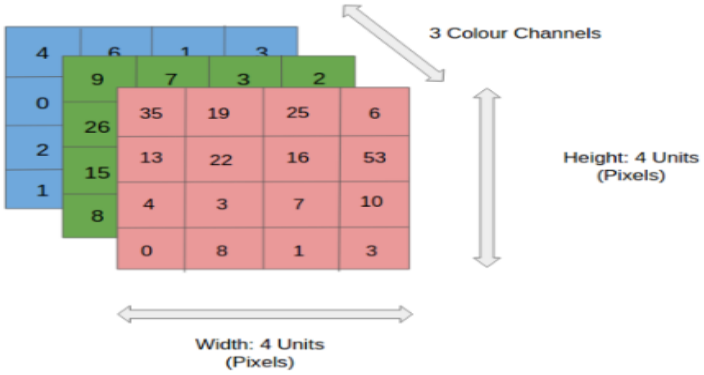


Fig.10: RVB Image

Pour plus de simplicité, restons-en aux images en niveaux de gris alors que nous essayons de comprendre le fonctionnement des CNN.

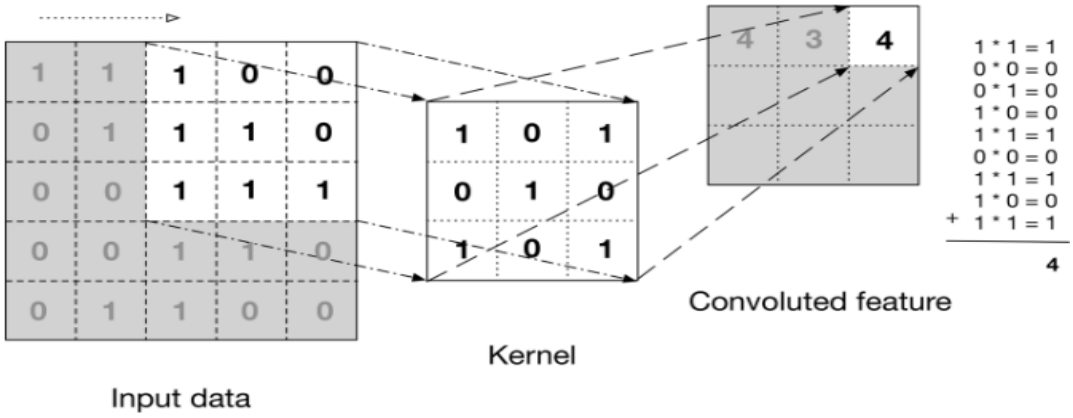


Fig.11: Input Black and white picture

L'image ci-dessus montre ce qu'est une convolution. Nous prenons un filtre/noyau (matrice 3 × 3) et l'appliquons à l'image d'entrée pour obtenir la fonction convoluée. Cette fonction convoluée est transmise à la couche suivante. Dans le cas de la couleur RVB, regarde la figure ci-dessous pour comprendre son fonctionnement :

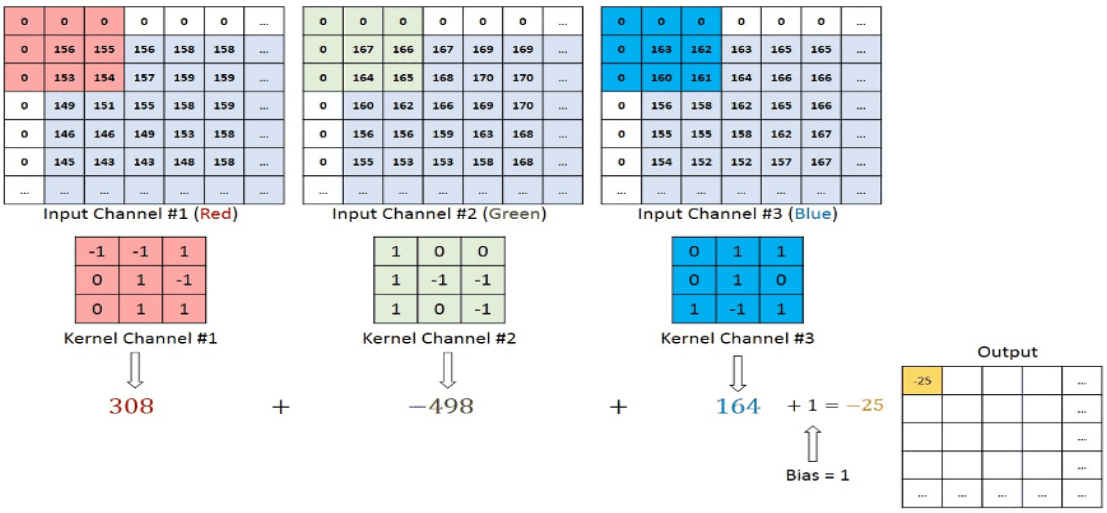


Fig.12: Input with RVB picture

Les réseaux de neurones convolutifs sont composés de plusieurs couches de neurones artificiels. Les neurones artificiels, une imitation grossière de leurs homologues biologiques, sont des fonctions mathématiques qui calculent la somme pondérée de plusieurs entrées et sorties d'une valeur d'activation. Lorsque vous entrez une image dans un ConvNet, chaque couche génère plusieurs fonctions d'activation qui sont transmises à la couche suivante. La première couche extrait généralement les caractéristiques de base telles que les bords horizontaux ou diagonaux. Cette sortie est transmise à la couche suivante qui détecte des caractéristiques plus complexes telles que des coins ou des arêtes combinatoires. Au fur et à mesure que nous progressons dans le

réseau, il peut identifier des caractéristiques encore plus complexes telles que des objets, des visages, etc.

Fig.13: Layers CNN

Sur la base de la carte d'activation de la couche de convolution finale, la couche de classification génère un ensemble de scores de confiance (valeurs comprises entre 0 et 1) qui spécifient la probabilité que l'image appartienne à une "classe". Par exemple, si nous avons un ConvNet qui détecte les chats, les chiens et les chevaux, la sortie de la couche finale est la possibilité que l'image d'entrée contienne l'un de ces animaux.

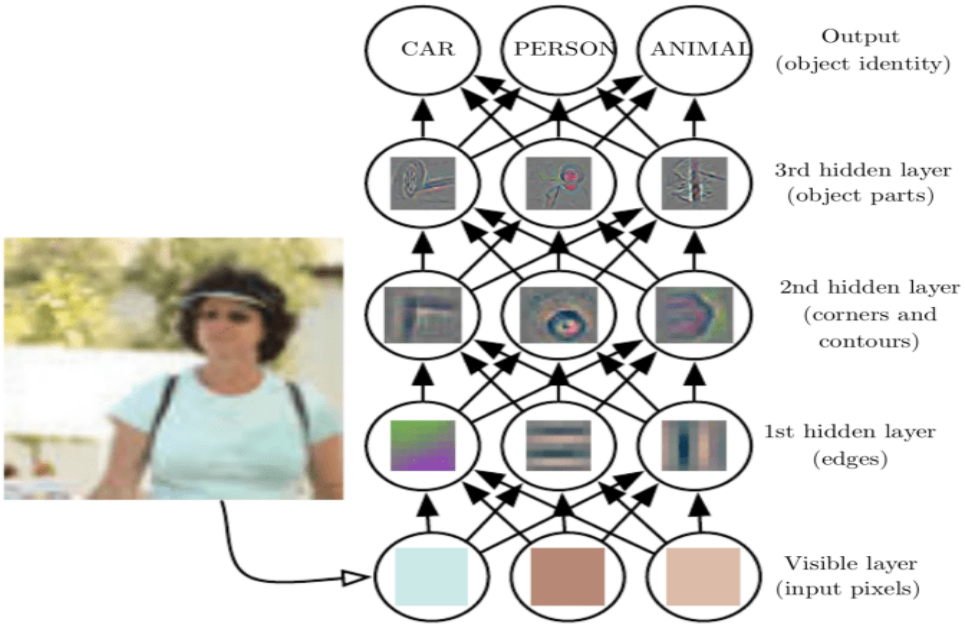


Fig.14: Prediction with CNN

- **Qu'est-ce qu'une couche de regroupement ?**

  Semblable à la couche convolutive, la couche de regroupement est responsable de la réduction de la taille spatiale de l'entité convoluée. Il s'agit de **diminuer la puissance de calcul nécessaire pour traiter les données** en réduisant les dimensions. Il existe deux types de mise en commun, la mise en commun moyenne et la mise en commun maximale. Donc, ce que nous faisons dans **Max Pooling**, c'est que nous trouvons la valeur maximale d'un pixel à partir d'une partie de l'image couverte par le noyau. **Max Pooling** fonctionne également comme un **suppresseur de bruit**. Il supprime complètement les activations bruyantes et effectue également un débruitage ainsi qu'une réduction de la dimensionnalité. D'autre part, **Average Pooling** renvoie la **moyenne de toutes les valeurs** de la partie de l'image couverte par le noyau. La mise en commun moyenne effectue simplement une réduction de la dimensionnalité en

tant que mécanisme de suppression du bruit. Par conséquent, nous pouvons dire que **Max Pooling est beaucoup plus performant qu'Average Pooling**.

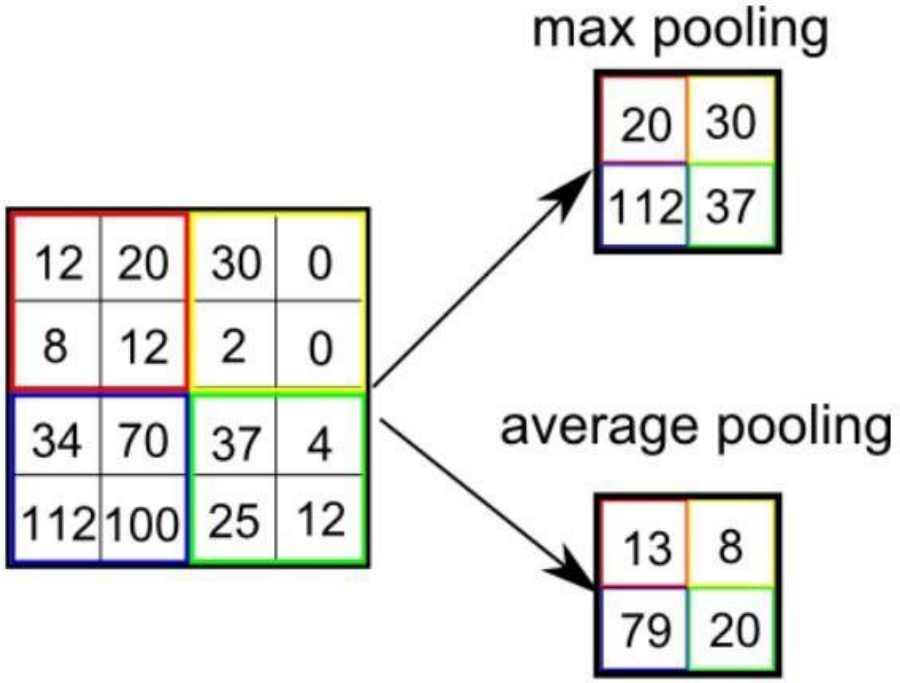


Fig.15: Max Pooling Versus Average Pooling

- **Limites de CNN :**

Malgré la puissance et la complexité des ressources des CNN, ils fournissent des résultats approfondis. À la base de tout cela, il s'agit simplement de reconnaître des motifs et des détails si infimes et si discrets qu'ils passent inaperçus à l'œil humain. Mais quand il s'agit de **comprendre** le contenu d'une image, cela échoue. Ces limites sont plus qu'évidentes lorsqu'il s'agit d'applications pratiques. Mais malgré les vastes ressources d'images et de vidéos sur lesquelles ils ont été formés, il n'est toujours pas en mesure de bloquer et de supprimer complètement le contenu inapproprié. Plusieurs études ont montré que les CNN formés sur ImageNet et d'autres ensembles de données populaires ne parviennent pas à détecter les objets lorsqu'ils les voient sous différentes

conditions d'éclairage et sous de nouveaux angles. Malgré les limites des réseaux de neurones convolutifs, force est de constater qu'ils ont révolutionné l'intelligence artificielle. Aujourd'hui, les CNN sont utilisés dans de nombreuses **applications de vision par ordinateur** telles que la reconnaissance faciale, la recherche et l'édition d'images, la réalité augmentée, etc. Comme le montrent les progrès des réseaux de neurones convolutifs, nos réalisations sont remarquables et utiles, mais nous sommes encore très loin de **reproduire les composants clés de l'intelligence humaine .**

- **Un extrait de code permettant de construire un modèle CNN avec Keras :**

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten#create model
model = Sequential()#add model layers
model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

3. Réseaux de neurones récurrents (Recurrent Neural Networks)

Dans la section précédente, nous avons découvert les réseaux de neurones convolutifs, qui sont modèles d'apprentissage profond supervisé qui ont révolutionné le domaine de la vision par ordinateur, en particulier la détection d'objets dans les images. Dans cette section, nous découvrirons un autre modèle d'apprentissage profond supervisé, qui est le modèle **réseau neuronal récurrent.** Jusqu'à présent, nous avons vu des réseaux de neurones et des modèles d'apprentissage en profondeur qui voient les points de données comme des instances indépendantes. Cependant, disons que nous voulons construire

un modèle capable d'analyser les scènes d'un film. Eh bien, nous ne pouvons pas supposer que les scènes d'un film sont indépendantes et, par conséquent, les modèles traditionnels d'apprentissage en profondeur ne conviennent pas à cette application. Les **réseaux de neurones récurrents** surmontent ce problème. **Réseaux de neurones récurrents ou (RNN)** pour faire court, sont des réseaux avec des boucles qui ne prennent pas simplement une nouvelle entrée, mais à la fois aussi prendre en entrée la sortie du point de données précédent qui a été alimenté dans le réseau. Donc, c'est ainsi que l'architecture d'un **réseau de neurone récurrent** ressemblerait (figure ci-dessous).

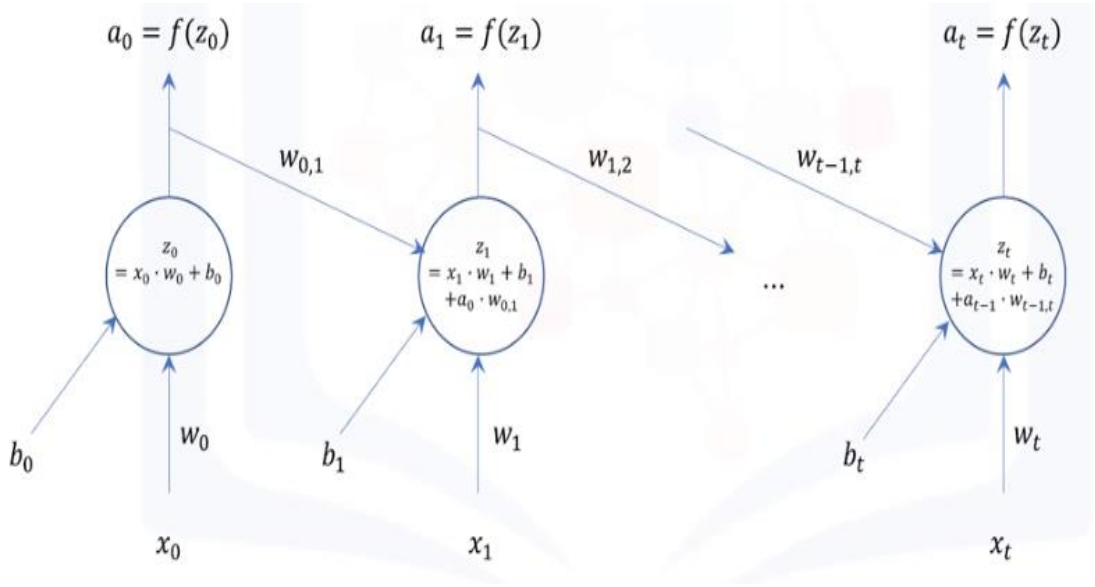


Fig.16: Recurrent Neural Networks

A l'instant t = 0, le réseau prend en entrée x0 et en sortie a0. Alors, au temps t = 1, en plus de l'entrée x1, le réseau prend également a0 en entrée, pondéré avec le poids w0,1, et ainsi de suite.

En conséquence, les réseaux de neurones récurrents sont très bons pour modéliser des modèles et des séquences de données, tels que les textes l'écriture

manuscrite et les marchés boursiers. Ces algorithmes prennent du temps et compte de la séquence, ce qui signifie qu'ils ont une dimension temporelle. Un type très populaire de réseau de neurones récurrent est le modèle de mémoire à long terme ou le modèle (long short-term model) ou (**LSTM**) en abrégé. Il a été utilisé avec succès pour de nombreuses applications y compris la génération d'images, où un modèle entraîné sur de nombreuses images est utilisé pour générer des images nouvelles. Une autre application est la génération d'écriture manuscrite. Aussi, les modèles **LSTM** ont été utilisés avec succès pour construire des algorithmes qui peuvent décrire automatiquement les images ainsi que les flux de vidéos.

4. Encodeurs automatiques (Autoencoders)

Jusqu'à présent, nous avons discuté de deux modèles d'apprentissage profond supervisé, qui sont les réseaux de neurones convolutifs et les réseaux de neurones récurrents. Dans cette section, nous passerons en revue un modèle d'apprentissage en profondeur non supervisé qui est **l'auto-encodeur**. **Que sont donc les encodeurs automatiques ?** L'encodage automatique est un algorithme de compression de données où les fonctions de compression et de décompression sont apprises automatiquement à partir des données au lieu d'être conçu par un humain. Ces auto-encodeurs sont construits en utilisant les réseaux de neurones. Les encodeurs automatiques sont spécifiques aux données, ce qui signifie qu'ils ne sont capables de compresser que des données similaires à celles sur lesquelles ils ont été formés. Par conséquent, un encodeur automatique formé sur des images de voitures ferait un travail plutôt médiocre

de compression d'images de bâtiments, car les caractéristiques qu'il apprendrait seraient spécifiques au véhicule ou à la voiture.

Quelques applications intéressantes des auto-encodeurs sont le débruitage des données et la réduction de la dimensionnalité des données.

Voici l'architecture d'un auto-encodeur.

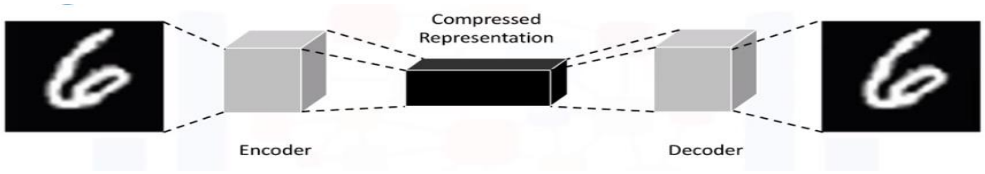


Fig17. : Autoencoders

Il faut une image, pour exemple, comme entrée et utilise un encodeur pour trouver la compression optimale de représentation de l'image d'entrée. Ensuite, à l'aide d'un décodeur, l'image originale est restaurée. Un auto-encodeur est donc un modèle de réseau de neurones non supervisé. Il utilise la rétropropagation en définissant la variable cible. En d'autres termes, il essaie d'apprendre une approximation d'une fonction identité. En raison des fonctions d'activation non linéaires dans les réseaux de neurones, les encodeurs automatiques peuvent apprendre des projections de données qui sont plus intéressantes par exemple l'analyse en composantes principales **PCA** ou autres techniques de base, qui peuvent traiter uniquement des transformations linéaires. Un type d'auto-encodeurs très populaire est le Machines Boltzmann restreintes ou (**RBM**) en abrégé. Les RBM ont été couronnés de succès et utilisées pour diverses applications, y compris la correction d'ensembles de données déséquilibrés (**Fixing Inbalance dataset**). Les **RBM** apprennent

l'entrée afin de pouvoir le régénérer, alors ils pourront apprendre la répartition de la classe minoritaire dans un ensemble de données déséquilibre, puis générer plus de points de données de cette classe, en transformant l'ensemble de données déséquilibre en un ensemble de données équilibré .De même, les RBM peuvent également être utilisés pour estimer les valeurs manquantes dans différents caractéristiques d'un ensemble de données. Une autre application populaire de **RBM** est une extraction automatique de caractéristiques de données particulièrement non structurées.

5. Pratique

(Pratice)

Ici, nous apprendrons à utiliser la bibliothèque Keras pour créer des réseaux de neurones convolutionnels. Cliquer sur le lien ci-dessous en accédant au notebook dans github et l'ouvrir dans **colab** :

https://github.com/gackouhamady/ConvolutionalNeuralNetworkPratice/blob/main/DL0101EN_4_1_Convolutional_Neural_Networks_with_Keras_py_v1_0__1_.ipynb

V. Projet Final

(Final Project)

Contexte :

Dans ce projet, nous construirons un modèle de régression à l'aide de la bibliothèque d'apprentissage en profondeur Keras, puis nous expérimenterons l'augmentation du nombre d'époques (époches) de formation et la modification du nombre de couches cachées et nous verrons comment la modification de ces paramètres affecte les performances du modèle. Pour la commodité, les données peuvent être retrouvées ici : https://cocl.us/concrete_data . Pour récapituler, les prédicteurs dans les données de résistance du béton comprennent : Cement (ciment), Blast Furnace Slag (Laitier de haut fourneau), Fly Ash (Cendres Volantes), Water (eau), Superplasticizer (superplastifiant), Coarse Aggregate (Agrégat grossier), Fine Aggregate (Agrégat de fin).

A. Construire un modèle de référence

Utilisez la bibliothèque Keras pour créer un réseau de neurones avec les éléments suivants :

- Une couche cachée de 10 nœuds et une fonction d'activation ReLU
- Utilisez l'optimiseur **Adam** et l'**erreur quadratique moyenne** comme fonction de perte.

1. Séparez aléatoirement les données en ensembles d'apprentissage et de test en conservant 30 % des données pour les tests. Vous pouvez utiliser la fonction d'assistance **train_test_split** de Scikit-learn.

2. Entraînez le modèle sur les données d'apprentissage en utilisant **50 époques**.

3. Évaluez le modèle sur les données de test et calculez l'erreur quadratique moyenne entre la résistance prédite du béton et la résistance réelle du béton. Vous pouvez utiliser la fonction **mean_squared_error** de Scikit-learn.

4. Répétez les étapes 1 à 3, **50 fois**, c'est-à-dire créez une liste de **50** erreurs quadratiques moyennes.

5. Indiquez la **moyenne et l'écart type des erreurs quadratiques moyennes.**

B. Normaliser les données

Répétez la partie A **mais utilisez une version normalisée des données**. Rappelez-vous qu'une façon de normaliser les données consiste à soustraire la moyenne des prédicteurs individuels et à la diviser par l'écart type. **Comment la moyenne des erreurs quadratiques moyennes se compare-t-elle à celle de l'étape A ?**

C. Incréer le nombre d'époques

Répétez la partie B **mais utilisez 100 époques cette fois pour l'entraînement. Comment la moyenne des erreurs quadratiques moyennes se compare-t-elle à celle de l'étape B ?**

D. Augmenter le nombre de couches cachés

Répétez la partie B, mais utilisez plutôt un réseau de neurones : Trois couches cachées, chacune de 10 nœuds et fonction d'activation ReLU. **Comment la moyenne des erreurs quadratiques moyennes se compare-t-elle à celle de l'étape B ?**

2. Concrétisation :

Cliquer sur le lien ci-dessous en accédant au notebook du projet concrétisé dans github et l'ouvrir dans **colab** :

https://github.com/gackouhamady/FinalProject_ML_NeuralNetworks_ENI-ABT_2023/blob/main/Deep_Learning_with_Keras_Assignment.ipynb

Conclusion Générale

Récapitulons ce que nous avons couverts dans cette présentation**. Dans la partie1**, nous avons passé en revue un certain nombre de choses passionnantes et quelques applications motivantes de l'apprentissage en profondeur. Nous avons également couvert brièvement les réseaux de neurones dans le cerveau pour apprécier comment ils inspirent les réseaux de neurones artificielle, puis nous avons appris comment les réseaux de neurones font des prédictions à travers le processus de propagation vers l'avant. **Dans la partie 2**, nous avons appris le gradient de descende et rétropropagation. **Dans la partie 3**, nous avons découvert des bibliothèques d'apprentissage en profondeur et en particulier Keras. **Dans la partie 4**, nous avons découvert les réseaux de neurones profonds supervisés et non supervisés, et nous avons utilisé la bibliothèque **Keras** pour construire un réseau de neurones convolutifs.

Enfin, Nous avons vraiment apprécié de realiser ce document ensemble, et donc nous espérons vraiment que ça vous a plu et que vous avez beaucoup appris de cela. C'était difficile pour nous, parce que nous avons dû laisser de côté beaucoup de détails pour faire simple. Le but de cette présentation n'était pas de vous apprendre tout sauf pour vous apprendre assez pour être prêt pour aborder des concepts plus avancés sur l'apprentissage en profondeur.

Merci pour avoir suivi et nous vous souhaitons tous des meilleures chances !

Bibliographies et webographies

Allibhai, E. (n.d.). Retrieved from https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

Coursera. (n.d.).

Durán, J. (n.d.). *Everything You Need to Know about Gradient Descent Applied to Neural Networks*. Retrieved from https://medium.com/yottabytes/everything-you-need-to-know-about-gradient-descent-applied-to-neural-networks-d70f85e0cc14

i2tutorials. (n.d.). Retrieved from https://www.i2tutorials.com/explain-deep-neural-network-and-shallow-neural-networks/#:~:text=Ans%3A%20Shallow%20neural%20networks%20give,built%20using%20various%20hidden%20layers.

Kwiatkowski, R. (n.d.). Retrieved from https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

Mandal, M. (n.d.). Retrieved from https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Singh, G. (6 septembre 2021). *Introduction to Artificial Neural Networks*. Retrieved from https://www.analyticsvidhya.com/blog/2021/09/introduction-to-artificial-neural-networks/

Singh, L. (n.d.). *Forward and Backward Propagation — Understanding it to master the model training process*. Retrieved from https://medium.com/geekculture/forward-and-backward-propagation-understanding-it-to-master-the-model-training-process-3819727dc5c1

Stanford Computer Science. (n.d.). *Neural Networks - Biology - Stanford Computer Science*. Retrieved from https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Biology/index.html