Introduction à la reconnaissance optique de caractère

Émile Bergeron, Samuel Paquin, Étienne Parent, Jérémie Sanfaço

nov. 27, 2020

Table des matières

La documentation sur l'utilisation de ce site n'a pas encore été faite. :)

CHAPITRE 1

Rapport Final

Cette section contient notre rapport final.

1.1 Introduction

1.1.1 Mise en contexte

L'intelligence artificielle est au coeur de l'actualité depuis près d'une décennie. Elle est déjà entrain de changer le monde , et ce, dans plusieurs secteurs incluant la finance, la sécurité, la santé, la justice criminelle, les moyens de transport, la publicité, et plusieurs autres.

Que ça soit des décisions sur l'investissement d'un portefeuille d'un individu ou de la détection de fraude en identifiant des anormalités, l'intelligence artificielle est de plus en plus présente dans le secteur de la finance. [Popper, 2016]

Du côté de la sécurité, un excellent exemple serait Project Maven un projet d" intelligence artificielle du Pentagon des États-Unis qui est capable de passer à travers plusieurs informations, vidéos et photos pour détecter des dangers potentiels.

L'intelligence artificielle est très importante dans la santé avec des compagnies comme Merantix, une compagnie allemande qui a permis de détecter des ganglions lymphatiques ainsi que des problèmes liés à ceux-ci tels que des lésions ou des cancers. L'étude de séquence d'ADN par l'intelligence artificielle permet de détecter des maladies génétiques et des cancers.

Un des domaines le plus importants en ce moment serait, les moyens de transport avec plus de $80 milliards investis dans des véhicules de conduite autonome entre 2014 et 2017. L'intelligence artificielle dans ce domaine aurait pour but de diminuer grandement l'erreur humaine dans les transports et réduire à presque zéro les accidents si la majorité des autos était intelligente. De plus, cela réduirait aussi grandement le trafic grâce à la communication entre les automobiles intelligentes. La compagnie Tesla en est déjà très avancée pour ce qui est de leur auto intelligente. [InnovationQuebec, 2018]

Comme on peut le voir, cette technologie a permis de multiples avancées dans des domaines où il se fait extrêmement difficile de modéliser la problématique selon une fonction mathématique particulière. L'analyse de langage en est un bon exemple. Le travail ne peut être modélisé par une seule fonction mathématique puisque les conditions souvent changeantes nécessiteraient une multitude de fonctions différentes pour chaque environnement qui n'est pas

réaliste. La solution est plutôt « d'entraîner » un ordinateur à comprendre le monde qui l'entoure. Pour continuer avec l'exemple de l'analyse du langage, une solution serait de fournir à l'ordinateur une immense quantité d'exemples et de solutions afin qu'il développe la capacité de prédire la solution à de nouveaux exemples. GPT-3, un nouveau modèle d'intelligence artificielle produit par OpenAI, a permis à des développeurs de créer un programme lui-même capable de programmer à partir de demandes spécifiques faites par un utilisateur.

1.1.2 Le début de la découverte des inconvénients

Malgré les avancées incroyables que l'intelligence artificielle a déjà permis et continuera de permettre dans le futur, elle n'est pas sans ses inconvénients.

Le biais

Au courant des dernières années, les systèmes intelligents sont de plus en plus reconnus coupables de discrimination envers certains groupes d'individus. Une étude réalisée par le NIST à étudié le taux d'erreur de différents programmes de reconnaissance faciale en fonction des différences de sexe et d'ethnicité des individus sur les photos analysées. L'étude présente des taux d'erreur jusqu'à cent fois plus élevés pour des personnes d'origine asiatique ou africaine lorsque comparé à des personnes d'origine européenne [Grother et al., 2019]. Le taux d'erreur est aussi plus élevé chez les femmes que chez les hommes, et ce, peut importe l'origine.

Un autre résultat important de cette étude est que le taux d'erreur associé à la reconnaissance de personnes asiatiques n'est pas présent dans des programmes réalisés dans des pays d'Asie. Cette observation permet de déduire l'un des plus grands problèmes liés à l'intelligence artificielle : le biais.

Contrairement à une fonction mathématique qui transforme un chiffre de manière définie, les procédés menant à la reconnaissance faciale sont beaucoup plus flous et souvent très mal compris. Plusieurs considèrent les programmes entraînés comme des « boîtes noires ». Il est difficile de prédire ce qui sortira de la boîte lorsque l'on y insère quelque chose, et il est encore plus difficile de comprendre pourquoi le programme prend certaines décisions plus que d'autres.

Cette imprévisibilité inquiète plusieurs. Elle rend la tâche de corriger le biais assez ardue. Elle fait aussi en sorte qu'il est difficile de prédire le comportement du programme dans des cas extrêmes sans avoir à lui faire passer des tests dans ces conditions. Le biais est donc un phénomène difficile à corriger, ce qui entraîne des questionnements en rapport aux bienfaits de l'utilisation de l'intelligence artificielle.

Certaines régions du monde commencent à bannir l'utilisation de la reconnaissance faciale par les forces de l'ordre. C'est le cas de la ville de Portland, en Oregon [Metz, 2020]. La ville a décidé de bannir l'utilisation de la technologie suite à des craintes en liées à son manque de précision, surtout lorsqu'utilisée sur des individus appartenant à une minorité visible.

Une deuxième révolution industrielle

Une autre inquiétude liée à l'intelligence artificielle est l'importante quantité d'emplois qui risque de disparaître puisqu'ils seront maintenant occupés par des ordinateurs. Ces inquiétudes sont justifiées. Plusieurs articles, dont celui-ci publié par CNN ainsi que cette publication sur Medium tentent de rassurer la population en mentionnant des emplois qui ne pourraient apparemment jamais être remplacés par des ordinateurs. Ils mentionnent entre autres les emplois créatifs, accompagnés des emplois nécessitant beaucoup d'interactions humaines.

Pourtant, le domaine de l'IA avance chaque année, et il existe maintenant une panoplie de programmes capable de composer de la musique, maîtriser les arts visuels ainsi qu"entretenir des conversations au téléphone.

Il est dangereux d'extrapoler le progrès qui a été fait au courant des dernières années sur les décennies à venir. Certaines lois limitant le développement de l'IA, ou des limitations physiques au présent rythme d'augmentation de la puissance de calcul des ordinateurs pourraient survenir grandement ralentir le développement de la technologie. Si nous tentons tout de même de le faire, les inquiétudes vécues par plusieurs semblent raisonnables.

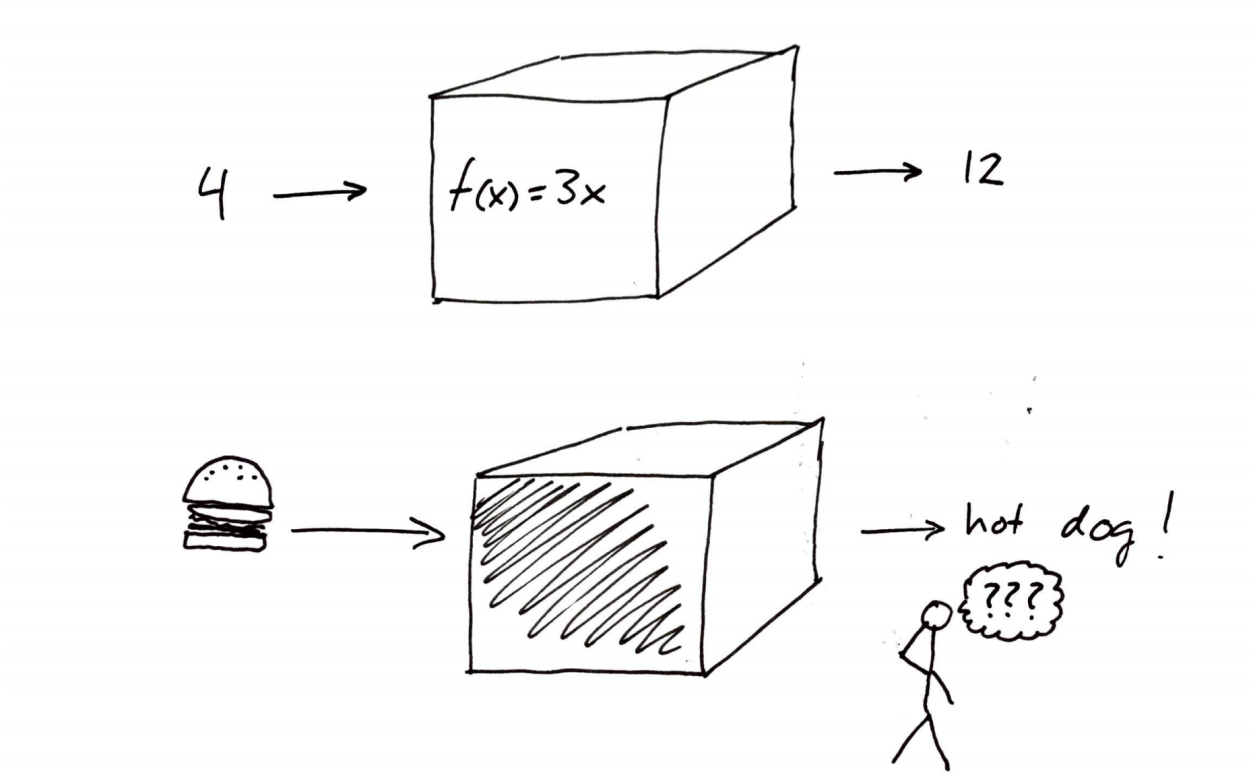


Fig. 1.1 – L'analogie de la boîte noire.

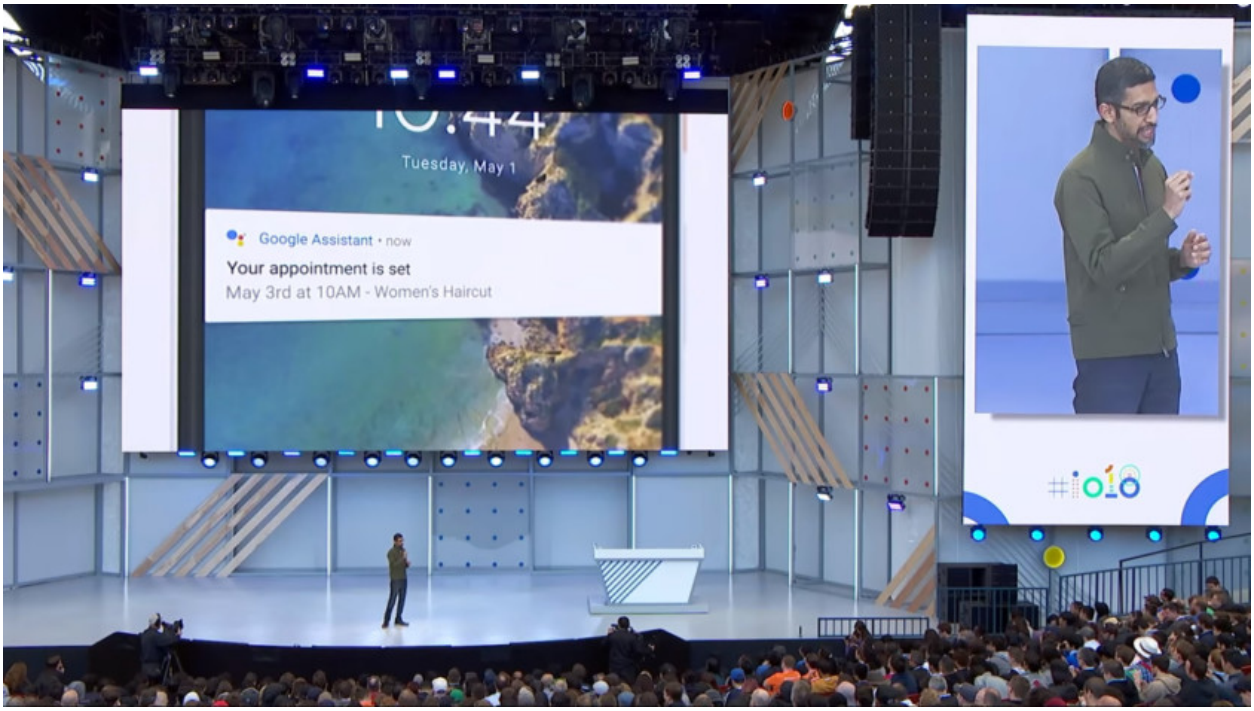


Fig. 1.2 – Le PDG de Google présentant une démonstration de Google Duplex.

1.1.3 Comprendre la technologie pour démystifier les inquiétudes

Bien que les précédentes inquiétudes face à l'intelligence artificielle soient totalement justifiées, elles ne sont pas sans solution. Si son développement est fait de manière éthique et s'il est bien encadré, nous pourrions en retirer plus d'avantages que d'inconvénient. Pour bien comprendre les inquiétudes, il faut d'abord comprendre les enjeux. C'est pourquoi nous tenterons de répondre à la question suivante.

Quel est le fonctionnent de l'intelligence artificielle et comment devrait-elle être utilisée afin de bénéficier l'être humain ?

1.2 Les librairies nécessaires

Afin de réaliser ce projet dans des temps raisonnables, nous utilisons des outils et des données réalisés par des organisations réputées comme Google, Numpy et la Matplotlib development team.

1.2.1 Tensorflow

Tensorflow est une plateforme nous permettant d'accélérer le développement de notre application d'apprentissage machine. La librairie procure un API en Python donnant accès à de multiples fonctions utilisées pour obtenir des données et les utiliser pour entraîner notre programme.

Historique

Développée par Google et rendue publique en 2015 [Les contributeurs de Wikipedia, 2020], la librairie a depuis permis aux masses de développer toutes sortes d'applications bénéficiant de l'intelligence artificielle. TensorFlow est une version polie du système DistBelief. DistBelief est un produit du projet The Google Brain. Après avoir été utilisé pendant quelques années pour des produits Google ainsi que pour de la recherche, DistBelief est amélioré et rendu publique sous le nom TensorFlow [The TensorFlow developper team, 2015]. Aujourd'hui, la page Github de TensorFlow mentionne plus de 100 000 utilisateurs et 2 780 contributeurs. La librairie est utilisée par de multiples entreprises dont Google, Coca-Cola, airbnb, Twitter et Intel [The TensorFlow developper team, 2020].

Notre utilisation

Nous utilisons TensorFlow afin de faciliter l'accès à nos données et afin de créer notre modèle.

Accès aux données

Pour entrainer notre modèle, nous utilisons la base de données MNIST. Bien qu'elle soit très complète, le format de cette base de donnée est assez complexe (*Voir la section sur le* preprocessing*)*. Heureusement, la libraire TensorFlow procure un interface simple avec le langage de programmation que nous utilisons.

```
((train_data, train_labels), (test_data, test_labels)) = mnist.load_data()
num_data = np.vstack([train_data, test_data])
num_labels = np.hstack([train_labels, test_labels])
```

La classe `mnist` fournie par la librairie `Tensorflow`nous permet de charger en mémoire toutes les données nécessaires en seulement trois lignes.

Entraînement du modèle

Le domaine de l'IA s'avère assez complexe. Programmer et entraîner un modèle nécessite des connaissances en mathématiques avancées [Goodfellow et al., 2016].

`Tensorflow`vise à accélérer le développement de l'intelligence artificielle ainsi que de rendre ce développement accessible aux masses. Afin de simplifier la réalisation de notre programme et afin de le rendre plus efficace, nous comptons donc utiliser les méthodes d'entraînement fournies par la librairie. Pour aider le lecteur à comprendre le fonctionnement du programme (et donc pour éviter le phénomène de la [boîte noire](lien vers la section de la boîte noire), chaque fonction utilisée sera décortiquée et expliquée en détails.

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])

model.compile(optimizer='adam',
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=10)
```

Seulement quelques lignes sont nécessaires à l'entraînement de notre modèle.

1.2.2 Numpy

NumPy est une librairie facilitant le calcul avec le langage de programmation que nous utilisons pour créer notre modèle.

Historique

La libraire à été créée en 2005 et était à l'époque basée sur les librairies numériques et les modules mathématiques de Python [The Numpy documentation team, 2020]. Numpy vise à rendre la réalisation de grands calculs numériques plus simples et optimisée. Plusieurs de ses capacités ont des fonctions homologue dans les logiciels Matlab et Maple.

Notre utilisation

Comme sera possible de l'observer tout au long de ce rapport, les mathématiques constituent le pilier principal sur lequel repose le domaine de l'intelligence artificielle. De plus, la plupart des composantes des réseaux neuronaux peuvent être représentés comme des matrices. Numpy permet d'utiliser certaines propriétés des matrices afin de paralléliser les calculs menant à l'entraînement du modèle. De plus, comme nous le verrons dans la section suivante, Numpy peut être utilisé afin de représenter des images comme des matrices, et donc faciliter les opérations sur chacun des pixels.

1.2.3 Matplolib

Matplotlib est une librairie de visualisation implémentée dans le langage de programmation Python.

Historique

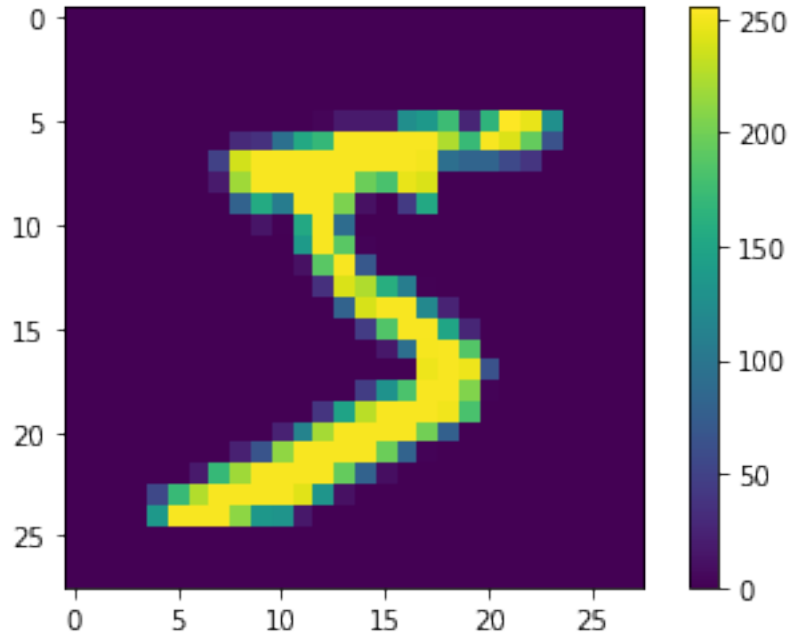
La libraire est un projet à code source ouvert financé par Numfocus. Déployé depuis plus de 17 ans citewikimatplotlib, la librairie permet, tout comme `Numpy`, au langage de programmation Python de se rapprocher de l'application Matlab, tout en restant libre de droit.

Notre utilisation

La librairie fourni un *API* simple à utiliser permettant de réaliser des graphiques directement dans nos notebooks. Nous utilisons aussi la fonctionnalité permettant de représenter des images comme suit :

```
plt.imshow(train_data[0])
```

Ce qui permet d'obtenir la figure suivante :



`train_data[0]` est un `array` de pixels, représentés par leur `grayscale value`. Nous discuterons plus de ces termes dans la section suivante.

1.3 Traitement antérieur à l'entrainement

Dans cette section, nous discuterons du traitement nécessaire afin d'utiliser des images pour entrainer un réseau neuronal. Nous discuterons aussi de l'importance de ce traitement, ainsi que de la raison pour laquelle il doit aussi être réalisé sur les images que nous voudrons par la suite reconnaître.

1.3.1 Pourquoi faire du *preprocessing* ?

Comme nous en avons discuté dans la section précédente, notre programme utilise des méthodes fournies par la librairie `Tensorflow` afin de charger les données dans le bon format. Malheureusement, dans une majorité des cas, les données ne vous seront pas fournies sur un plateau d'argent. Les programmes d'apprentissage machine visent à faire du calcul statistique sur le jeu de données fourni

---

Note : Dans le domaine de l'IA, un jeu de données représente l'ensemble des données traitées ainsi que leur étiquettes.

1.3.2 Mise en bouche sur l'apprentissage machine

L'entrainement d'un modèle se rapproche beaucoup des mathématiques, plus précisément de la statistique, comme en témoigne le *Deep Learning Book* [Goodfellow et al., 2016]. L'apprentissage machine vise à ingérer des quantités massives de données provenant de sources différentes. Par la suite, à l'aide de calculs statistiques, le programme tente de faire une certaine classification du jeu données. Selon le besoin, le programme pourrait alors poser une étiquette sur des données non étiquetées similaires à celles retrouvées dans le jeu de donnée [mitclassification]. Le modèle pourrait aussi être entrainer afin de reconnaître des anomalies, grouper des informations similaires par classes et bien d'autres [wikisupervised]. Toutes ces informations seront discutées plus en détails au courant de la prochaine section. Ce qu'il est important de retenir, c'est qu'entrainer un modèle nécessite ***beaucoup*** de données.

Sur la signification de «beaucoup de données»

Il y a 60 000 exemples dans nos données d'entraînement.

```
In [ ]: train_data.shape
Out[ ]: (60000, 28, 28)
```

1.3.3 Traiter beaucoup de données

Pour reprendre l'exemple précédent, la `shape` de l'objet `train_data`est une liste de 60 000 images représentées par des matrices carrées de dimension 28. Dans notre cas, si le programme passait tous les pixels un à un, il faudrait qu'il réalise séquentiellement $28 \times 28 \times 60000 = 47040000$ opérations. Ce serait par exemple le cas dans une `for loop`. Bien que les ordinateurs modernes sont particulièrement rapides [16], les modèles récents sont eux aussi entrainés avec des jeux de données de plus en plus massifs. Celui utilisé pour le service Google Translate, par exemple, compte des milliards d'exemples [googledatasize].

Heureusement, il existe des méthodes permettant de paralléliser [17] les opérations statistiques réalisées sur notre modèle.

16. Un ordinateur moderne possédant un processeur de 2GHz peut réaliser 2 000 000 000 opérations par seconde sur chacun de ses coeurs.
17. Exécuter plusieurs opérations parallèlement plutôt que séquentiellement.

Le parallélisme

Pour réduire le coût monétaire et temporel de l'entrainement d'un modèle, la tâche peut être séparée sur plusieurs des coeurs [18] de la machine. Pour se faire, nous profiterons des propriétés des matrices.

Les propriétés des matrices

Afin de trouver comment il serait possible de réaliser nos calculs en parallèle, analysons les propriétés des matrices.

La multiplication

Assumons les matrices de dimensions compatibles [19] A et B :

$$[A \times B]_{i,j} = \sum_{k=1} A_{i,k} B_{k,j}$$

Assumons aussi que la matrice C est produite par l'opération $A \times B$ et que les matrices A et B sont carrées [20]. Le calcul de C pourrait alors être implémenté de la manière suivante.

```
# Initialisation de la matrice de départ.
# Nous assumons que les matrices sont 3x3.
C = [[0,0,0],
    [0,0,0],
    [0,0,0]]
# Pour chaque rangée de la matrice A.
for i in range(len(A)):
        # Pour chaque valeur d'une rangée de la matrice B.
        for j in range(len(B[0])):
                # Pour chaque rangée de la matrice B
                for k in range(len(B)):
                        # [AxB]_{i,j} += A_{i,k} * B_{k, j}
                        C[i][j] += A[i][k] * B[k][j]
```

Quoi qu'assez simple à implémenter, cette façon de calculer C est particulièrement inefficace. Alors que les matrices A et B augmentent en taille, le nombre d'opérations requises augmente... **au cube !** Si A passe d'une matrice $2X2$ à une matrice $3X3$, chaque `for loop` doit être réalisée n [21] fois de plus. Comme le programme contient 3 for loops imbriquées, si la première doit être faite n fois de plus, alors c'est de même pour la deuxième, puis la troisième. Le calcul est alors $n \times n \times n = n^3$ fois plus complexe à réaliser [wikimatrixmulti].

Heureusement, ce problème n'est pas sans issues. Reprenons l'équation de la multiplication de deux matrices. $[A \times B]_{i,j} = \sum_{k=1} A_{i,k} B_{k,j}$ Dans ce cas, chaque élément de C est produit par une sommation sur des multiplications d'éléments de A et B. Il est aussi important de noter qu'aucun calcul pour un élément de C dépend d'un calcul pour un autre élément de C [22]. Il serait donc possible de calculer plusieurs éléments de C en même temps !

Bien que le calcul en parallèle ne réduit pas l'ordre de complexité, il permet tout de même de diviser le temps requis par le nombre de coeurs utilisés [23].

18. Un «coeur» est une unité du processeur pouvant faire du calcul indépendamment des autres coeurs. Un processeur d'ordinateur portable moderne possède de deux à quatre coeurs. Une carte graphique moderne en possède quelques milliers, quoique moins performants que ceux du processeur.
19. Pour réaliser une multiplication entre deux matrices, il faut que le nombre de colonnes de la première matrice soit égal au nombre de rangées de la deuxième.
20. Une matrice carrée est une matrice qui contient autant de rangées que de colonnes.
21. n représente le nombre de rangées et de colonnes d'une matrice carrée.
22. Ils peuvent être calculés dans n'importe quel ordre et le résultat sera toujours le même. Un résultat pourrait aussi
23. Plus ou moins. Voir 1.1 Amdahl's law and Gustafson's law

L'addition

L'addition de deux matrices compatibles [24] se définit par l'addition de chacun des éléments homologues des deux matrices. Si nous reprenons les matrices carrées A et B utilisées plus haut, la somme de ces deux matrices serait :

$[A + B]_{i,j} = A_{i,j} + B_{i,j}$

Supposons que la matrice C résulte de la somme de A et B. Il est encore une fois possible d'affirmer que la valeur de $C_{i,j}$ ne dépend pas de la valeur de $C_{k,l}$. Il serait possible d'additionner chaque composante des deux matrices dans n'importe quel ordre en obtenant toujours le même résultat.

Encore une fois, l'addition de deux matrices peut être parallélisé afin de réduire le temps de calcul [25].

La multiplication par un scalaire

Bien que la multiplication par un scalaire s'avère facile à réaliser à la main pour de petites matrices, l'opération doit tout de même être réalisée sur chaque élément de la matrice.

$$\lambda \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \dots & x_{dn} \end{bmatrix}$$

Revient à faire le calcul :

$$\begin{bmatrix} \lambda x_{11} & \lambda x_{12} & \lambda x_{13} & \dots & \lambda x_{1n} \\ \lambda x_{21} & \lambda x_{22} & \lambda x_{23} & \dots & \lambda x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda x_{d1} & \lambda x_{d2} & \lambda x_{d3} & \dots & \lambda x_{dn} \end{bmatrix}$$

Encore une fois, aucun résultat n'est dépendant d'un autre. Il serait donc possible d'effectuer plusieurs multiplications en même temps, puis grouper les résultats dans une matrice.

La sommation

Nous avons ici un calcul légèrement différent des autres. Dans le cas de la sommation des éléments d'une matrice de dimension $(1, n)$, le calcul s'avère commutatif en plus d'être associatif [26]. La commutativité de l'addition permet à notre programme d'utiliser l'opérateur de réduction.

L'opérateur de réduction

Un opérateur de réduction permet de réduire les éléments d'un tableau à un seul résultat. [wikireducop]

En premier lieu, voici comme une addition séquentielle d'un tableau pourrait être réalisé. Assumons un tableau de 8 entiers comme suit : `tableau = [2,9,6,4,1,3,8,8]`. L'addition pourrait alors être réalisée en ajoutant chaque nombre un par un jusqu'à obtenir le total.

```
# Création du tableau.
tableau = [2,9,6,4,1,3,8,8]
# Initiation de la variable `somme`.
```

(suite sur la page suivante)

24. Pour que deux matrices puissent être additionnées, elles doivent avoir le même nombre de rangées et de colonnes.
25. En assumant que le calcul est réalisé sur une machine possédant plusieurs coeurs.
26. Dans le cas des autres opérations matricielles présentées, elles étaient seulement associatives. Les calculs étaient, comme pour la sommation, indépendants les un des autres. Par contre, pour les autres opérations les résultats devaient être placés de manière ordonnée dans la matrice résultante.

(suite de la page précédente)

```
somme = 0
# Pour chaque chiffre dans le tableau.
for chiffre in tableau:
    # Sommation de l'ancienne somme avec le nouveau chiffre.
    somme += chiffre
# Affichage de la somme.
print(somme)
```

Cet exemple permettrait de réaliser une sommation séquentielle sur tous les chiffres contenus dans le tableau. Ce reviendrait à réaliser le calcul suivant : $(((((((2+9)+6)+4)+1)+3)+8)+8)$ Bien que la moindre performance de cette méthode ne se fait pas ressentir pour des petites sommations, ce programme ne s'adapte pas bien à de grands tableaux [27].

Ensuite, si l'addition n'était qu'associative, l'opération pourrait tout de même être parallélisée. Les sommes partielles pourraient être calculées indépendamment les unes des autres comme pour les autres opérations matricielles. Le calcul serait similaire à celui ci : $((2+1)+(9+3))+((6+8)+(4+8))$ Dans cet exemple, les sommes $2+1$, $9+3$, $6+8$ et $4+8$ sont calculées en même temps. Par contre, lorsque l'une des opérations est complétée avant une autre, il arrive que l'ordinateur ait à attendre [stackoverflowcommutativity]. Le coeur ne pourrait alors pas se libérer pour faire d'autres opérations. Par exemple, assumons un ordinateur possédant deux unités de calcul disponibles et une addition non commutative. Si le calcul de $2+9$ était complété avant celui de $9+3$, l'ordinateur devrait attendre que les deux calculs soient complétés avant de calculer la somme partielle $(2+9)+(9+3)$. Heureusement, l'addition est associative. L'ordinateur ira donc écrire le résultat de la première opération complétée à la somme partielle, sans se soucier de la complétion de l'autre opération. L'unité de calcul sera alors libérée pour calculer, par exemple, la somme $6+8$.

Finalement , l'opérateur de réduction est beaucoup plus adapté aux échelles de l'intelligence artificielle. Encore une fois, l'ordinateur sépare la sommation en plusieurs petites opérations qui peuvent être exécutés en parallèle. De plus, l'opérateur profite de l'associativité pour optimiser la tâche au maximum. À la fin de la réduction, il ne reste qu'une seule addition. Le calcul mathématique serait par contre identique au précédent. $((2+1)+(9+3))+((6+8)+(4+8))$ Il est possible de remarquer que l'addition est fait dans un ordre particulier. Cet ordre donne un meilleur modèle d'accès à la mémoire.

Python utilise l'opérateur de réduction lors du calcul de sommations. Implémenter ce genre de solution s'avère donc assez simple.

```
tableau = [2,9,6,4,1,3,8,8]
somme = sum(tableau)
print(somme)
```

`NumPy` possède aussi une fonction de sommation optimisée pour les `numpy arrays`. Elle peut être implémentée tout aussi simplement.

```
import numpy as np
tableau = np.array([2,9,6,4,1,3,8,8])
somme = np.sum(tableau)
print(somme)
```

27. Voir la {section} sur les résultats concrets.

En bref

En bref, une majorité des opérations matricielles peuvent être parallélisées. Les matrices sont donc la représentation de choix pour les jeux de données dans le domaine de l'intelligence artificielle. La transformation du jeu de données en matrices est une partie majeure du *preprocessing*. Elle permet d'accélérer le calcul d'un facteur non-négligeable.

NumPy

C'est pour les opérations parallèles que la librairie `numpy`, mentionnée lors de la section précédente, entre en jeu. Les opérations matricielles réalisées à l'aide de méthodes implémentés par `numpy`profitent aussi de l'implémentation des `BLAS` [28]. Les `BLAS` permettent de grandement accélérer nos calculs sans même nécessiter de coeurs supplémentaires. Elles exploitent plutôt les différentes architectures de processeur ainsi que leur différents niveau de cache [29].

BLAS

Les sous-routines d'algèbre linéaire permettent de nettement réduire l'ordre de complexité des opérations d'algèbre linéaire. Elles permettent, par exemple, de décomposer des matrices en blocs afin d'accélérer la multiplication.

Ces sous-programmes sont extrêmement populaires. Ils sont implémentés dans une majorité des programmes de calcul scientifique [blaswebsite].

Quelques résultats concrets

Voici quelques résultats plus concrets permettant d'obtenir une meilleure idée de l'ampleur de l'accélération des calculs.

Multiplication de matrices à l'aide de `NumPy`.

Dans ce programme, deux matrices de dimensions $1000x1000$ sont multipliées. La méthode de base avec les itérations imbriquées est comparée avec l'implémentation de l'opération par la librairie `NumPy`.

```
import time
import numpy as np

# Création de nos matrices
A = np.random.rand(1000,1000)
B = np.random.rand(1000,1000)
C = np.zeros((1000,1000))

# Test de la première implémentation

start_time = time.time()

for i in range(len(A)):
        for j in range(len(B[0])):
                for k in range(len(B)):
                #         print(A[i][k])
#                         print(B[k][j])
#                         print(i,j)
```

(suite sur la page suivante)

28. `BLAS` signifie **B**asic **L**inear **A**lgebra **S**ubroutines, ou sous-routines de base d'algèbre linéaire.
29. Petite mémoire rapide allouée au processeur.

(suite de la page précédente)

```
                        C[i][j] += A[i][k] * B[k][j]

end_time = time.time()

time_1 = end_time - start_time

# Test de l'implémentation avec Numpy
start_time = time.time()

C = A*B

end_time = time.time()

time_2 = end_time - start_time

print("Run time 1 = {} seconds".format(time_1))
print("Run time numpy = {} seconds".format(time_2))
```

```
Run time 1 = 1838.9336512088776 seconds
Run time numpy = 0.005700111389160156 seconds
```

Alors que l'implémentation de base prend plus de 30 minutes à faire le calcul, `Numpy` n'a besoin de que moins de 6 millièmes de secondes [30] !

Sommations de tableaux à l'aide de `NumPy`

Dans cet autre programme démontre quant à lui la différence entre notre implémentation de base de la sommation avec celle de `NumPy` sur un `numpy array`. La sommation se fait sur 1 milliard d'éléments aléatoires entre 0 et 1.

```
import time
import numpy as np

# Création du tableau et initiation de la somme.
tableau = np.random.rand(1000000000)
somme = 0

# Test de la première implémentation.

start_time = time.time()

# Pour chaque chiffre dans le tableau.
for chiffre in tableau:
    # Sommation de l'ancienne somme avec le nouveau chiffre.
    somme += chiffre

end_time = time.time()

time_1 = end_time - start_time

# Test de l'implémentation avec Numpy.

# Réinitialisation de la somme.
somme = 0
```

(suite sur la page suivante)

30. Testé sur un processeur *2.9 GHz Dual-Core Intel Core i5*.

(suite de la page précédente)

```
start_time = time.time()

somme += np.sum(tableau)

end_time = time.time()

time_2 = end_time - start_time

print("Run time 1 = {} seconds".format(time_1))
print("Run time numpy = {} seconds".format(time_2))
```

La différence entre les résultats est encore une fois majeure.

```
Run time 1 = 361.87627387046814 seconds
Run time numpy = 5.082181215286255 seconds
```

1.3.4 Représenter des images

Pour notre programme, l'intrant de notre réseau neuronal sera constitué d'images. Par contre, ces images ne seront pas directement passées au travers de notre programme dans leur format d'origine.

Comme discuté dans la section précédente, il serait préférable que les images soient représentées sous forme de matrices. C'est heureusement déjà le cas de nos données d'entrainement.

```
In [ ]: type(train_data)
Out[ ]: numpy.ndarray
```

Dans cet exemple, `train_data` est un `array` contenant l'ensemble de nos images. Pour obtenir le nombre d'éléments dans cet `array`, la méthode `shape` peut être utilisée.

```
In [ ]: train_data.shape
Out[ ]: (60000, 28, 28)
```

La première valeur correspond au nombre d'image dans nos données d'entrainement. Les deux valeurs suivantes sont le nombre de rangées et de colonnes des matrices utilisées pour représenter ces mêmes images. Ce sont donc des matrices carrées de dimensions $n = 28$.

Afin d'analyser précisément l'une de ces images, imprimons l'une des rangées de pixels.

```
In [ ]: train_data[0][20]
Out[ ]: array([  0,   0,   0,   0,   0,
                 0,   0,   0,   0,   0,
                24, 114, 221, 253, 253,
                253, 253, 201,  78,   0,
                0,   0,   0,   0,   0,   0,
                0,   0], dtype=uint8)
```

Les pixels sont représentés par des valeurs *grayscale* inversées. Traditionnellement, une valeur *grayscale* est élevée lorsque le pixel est très illuminé. La valeur maximale de `255`signifie un blanc, alors que le `0`correspond au noir. Dans notre jeu de données, ces deux valeurs sont inversées. Une valeur élevée signifie un pixel plus sombre.

Le paramètre `dtype=uint8`signifie que nos pixels sont représentés par des entiers de 8 bits. Chaque bit ne pouvant avoir une valeur que de 1 ou 0, le plus grand nombre pouvant être représenté par ce type d'entier est 255.

Explications plus détaillées sur la représentation des images.

Cette section c'est seulement si j'ai le temps.

Pourquoi le *grayscale* ?

Pourquoi est-il inversé ?

Pourquoi seulement 784 pixels ?

1.4 Notions de base

1.4.1 Introduction au réseau neuronal

Un réseau neuronal est une forme d'intelligence artificielle, qui effectue des prédictions basées sur des valeurs qui sont entrées dans le système, afin d'accomplir une certaine tâche. Le réseau est constitué d'un ensemble de neurones interconnectés et distribués en plusieurs couches.

Chaque neurone possède des paramètres qui peuvent être ajustés, afin d'obtenir des résultats plus fiables. C'est ce qu'on appele l'entrainement. Le réseau est entrainé à partir d'un jeu de données, qui contient des valeurs associées à une étiquette, qui consiste de la « réponse » attendue.

Par exemple, un réseau neuronal ayant comme objectif de prédire l'achalandage dans un parc d'amusement pour une journée donnée pourrait recevoir comme intrant la température, le niveau d'ensoleillement ainsi que le pourcentage de précipitation et d'humidité. Le jeu de données serait alors constituée d'une liste ces quatres valeurs enregistrées à chaque jour des dernières années, avec comme étiquette le nombre de clients cette journée-là. Les réponses du réseau sont comparées aux étiquettes, et les paramètres des neurones sont individuellement modifiés de manière à se rapprocher de la réponse attendue.

1.4.2 OCR

Le terme OCR, ou ROC en français, signifie « Reconnaissance optique de caractères ». Cela désigne un processus aucours duquel du texte est extrait d'une image ou d'un document afin d'être transformé en fichier. Pour ce faire, un réseau neuronal reçoit les valeurs des pixels du document de source,

> Note : La valeur d'un pixel en « grayscale » ou échelle de gris, est un nombre entier de format 8 bits et peut donc avoir une valeur comprise entre 0 et 255 (2^8 - 1), où 0 est noir et 255 est blanc. Un pixel en couleur est représenté sous la forme d'un vecteur de 3 nombres 8 bits, chaque nombre correspondant à une valeur de rouge, vert et bleu. [RFisher & Wolfart, 2003]

traitées afin de les rendre utilisables par le réseau. Ces données se propagent ensuite vers l'avant dans le réseau, de couche de neurone en couche de neurone, avant d'aboutir à la couche d'extrants, composée de 10 neurones dans le cas de notre programme, qui correspondent aux chiffres de 0 à 9. Un de ces neurones de cette couche finale s'active, donnant ainsi le résultat estimé par le réseau. Ensuite, divers paramètres sont ajustés par un algorithme d'optimisation afin d'augmenter la précision des réponses du réseau.

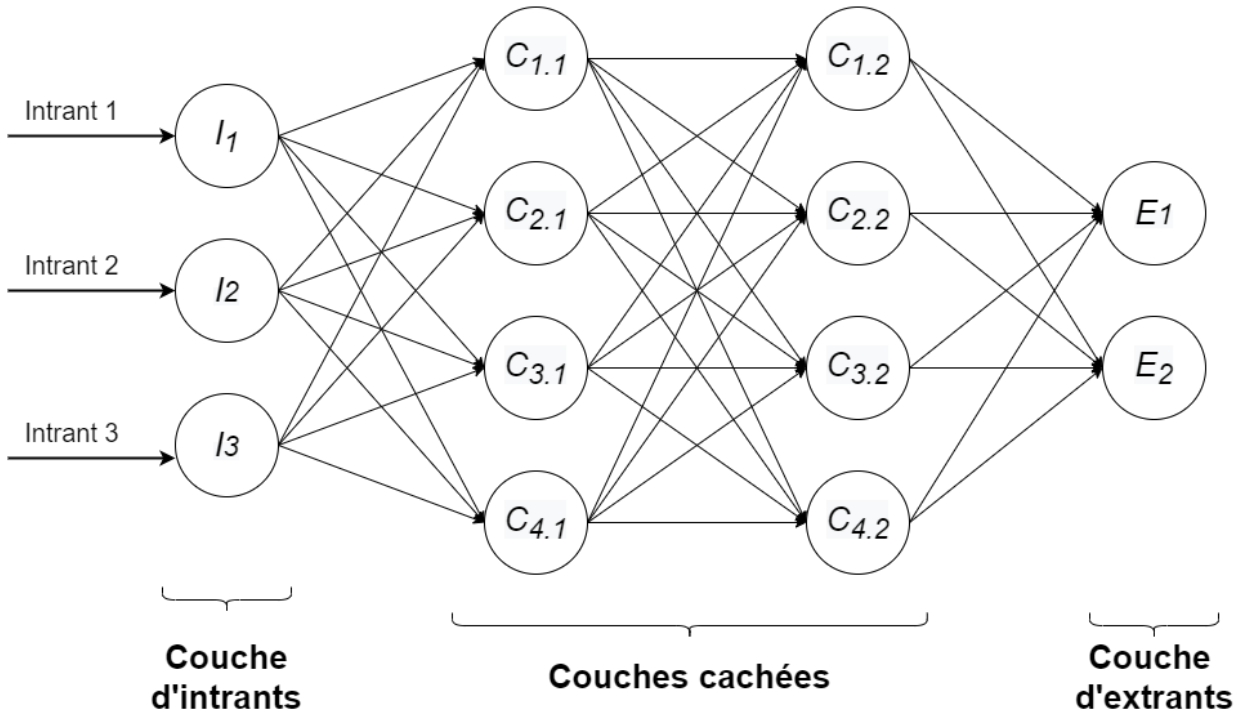


Fig. 1.3 – Ceci est un exemple simplifié d'un réseau neuronal. Les composantes du schéma seront expliquées en détail dans cette section.

1.4.3 Le neurone

Le neurone est l'unité de base d'un réseau neuronal. C'est un noeud parmis le réseau par lequel transitent des valeurs, qui sont modifiées au passage par un procédé qui sera expliqué plus en détail prochainement, avant d'être envoyées vers les prochains neurones. Essentiellement, un neurone reçoit une ou des valeurs comme intrant, effectue des opérations sur ces dernières, puis transmet la nouvelle valeur.

La structure d'un neurone est relativement simple. Chaque neurone possède un coefficient, ou un **poids** p dans le jargon, associé à chaque **intrant** I qu'elle reçoit. La première opération que la neurone effectue est la somme des produits des intrants fois leur poids. À celà est ajouté un **biais** b propre à chaque neurone. Cette opération peut être représentée par la fonction $Y = \sum_{i=1}^{n} I_i \times p_i + b$, où n correspond au nombre d'intrants.

La dernière opération que les valeurs subissent avant d'être transmises est une fonction d'activation. La fonction d'activation est appliquée à chaque extrant de chaque neurone de la couche. Les fonctions d'activation, analogues à l'activation d'un neurone biologique, permettent généralement d'obtenir un extrant compris entre 0 et 1, ou -1 et 1. Elles ont plusieurs utilités, notamment pour la modélisation de fonctions non linéeaires, ainsi que pour l'entrainement du réseau, ce qui sera expliqué dans une section ultérieure.

La fonction la plus simple est la fonction à échelons. Elle retourne 1 si l'intrant x est plus grand qu'une valeur seuil s, et 0 s'il ne l'est pas. Cette fonction peut être représentée par l'équation $E(x) = \begin{cases} 1 & \text{si } x > \text{s} \\ 0 & \text{si } x <= \text{s} \end{cases}$ Elle n'est néanmoins pas utilisée, puisqu'elle empêche l'entrainement du réseau. La fonction d'activation doit être dérivable en une autre fonction, et non en une constante, afin que le processus d'ajustement des paramètres puisse avoir lieu. Il est également impossible de représenter des situations non-linéeaires avec cette fonction, puisque seulement des fonctions linéaires sont présentes dans le réseau.

La fonction d'activation la plus utilisée est la fonction Unité Linéaire Rectifiée, ou « ReLU » en anglais (Rectified

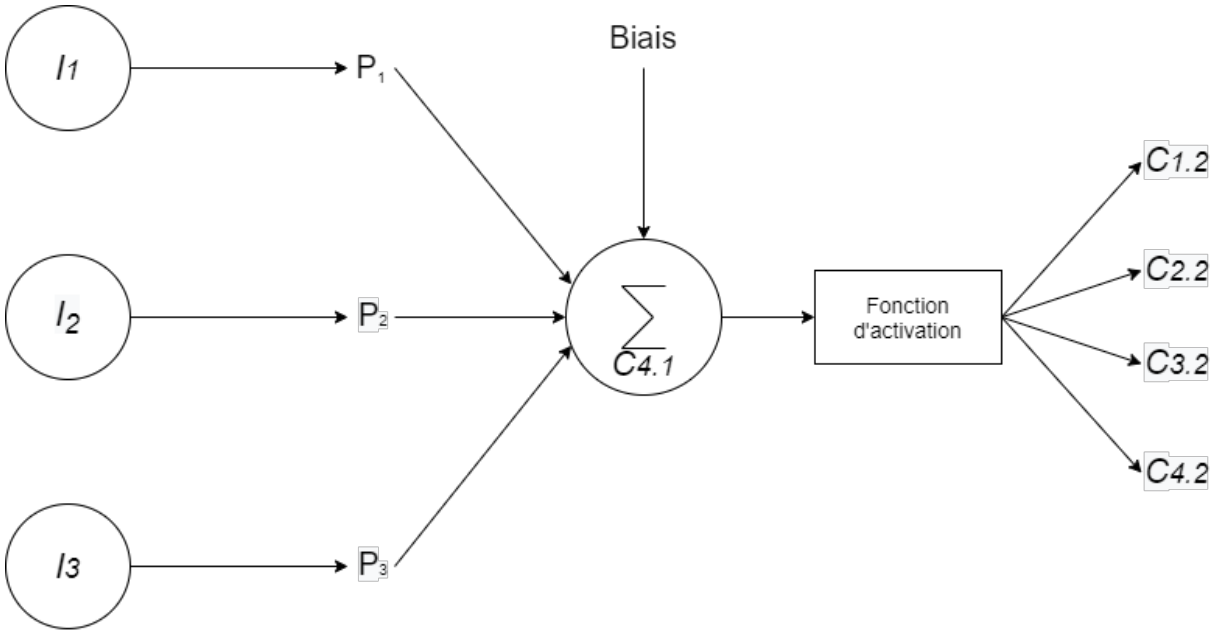


Fig. 1.4 – Exemple des opérations effectuées au sein d'un neurone.

Linear Unit). Cette fonction peut être représentée par l'équation : $R(x) = \begin{cases} x & \text{si } x > 0 \\ 0 & \text{si } x <= 0 \end{cases}$

ou encore, $R(x) = max(0, x)$. Cette fonction est peu demandante à calculer pour l'ordinateur, et se fait très rapidement. De plus, malgré son apparence linéaire, elle peut être dérivée, ce qui est nécessaire pour pouvoir entrainer le réseau. C'est pour ces raisons que c'est la fonction d'activation la plus répendue. Elle a toutefois comme désavantage de produire parfois une trop grande quantité de « 0 », ce qui peut entrainer une réaction en chaine, où ces zéros se propagent, empêchant le bon fonctionnement du réseau. Cette situation est appelée la « mort du réseau », où l'extrant de plusieurs neurones devient invariablement 0, ce qui diminue l'efficacité du réseau. Ce phénomène se produit surtout lorsque le réseau se fait entrainer de manière trop rigoureuse, et que le biais de certaines neurones devient une très grande valeur négative, ce qui fait que l'intrant dans la fonction d'activation est toujours en dessous de 0, et l'extrant reste ainsi invariablement 0.

Une variation de cette fonction, nommée Leaky ReLU, a été créée afin de tenter de régler ce problème de mort du réseau : $L(x) = \begin{cases} x & \text{si } x > 0 \\ 0,01 \times x & \text{si } x <= 0 \end{cases}$

Ici, les zéros sont remplacés par de très petits nombres négatifs, qui correspondent généralement à x multiplié par le coefficient 0,01.

Une autre fonction commune est la sigmoide. Son équation est : $\phi(x) = \frac{1}{1+e^{-x}}$ La fonction retourne 0 lorsque x tend vers l'infini négatif, et 1 lorsque x tend vers l'infini positif. Cette fonction a comme avantage de s'approcher rapidement de 0 ou de 1, lorsque l'intrant *x* est plus petit que -2 ou plus grand que 2, respectivement. Cela permet d'envoyer un signal très fort aux prochains neurones. Cela peut toutefois devenir un désavantage lorsque les intrants sont très grands, puisque l'extrant reste pratiquement le même, ce qui peut nuire à l'entrainement. Cette fonction est également plus lourde pour l'ordinateur, ce qui peut ralentir considérablement le système lorsque ce calcul est effectué des centaines ou des milliers de fois.

Une fonction similaire à la sigmoide et la TanH. Son équation est : $tanh(x) = \frac{2}{1+e^{-2x}} - 1$ Elle retourne -1 lorsque x tend vers l'infini négatif, et 1 lorsque x tend vers l'infini positif. Elle a comme avantage de retourner en moyenne des valeurs proches de 0, ce qui rend la tâche plus facile pour les couches suivantes, puisque les valeurs auront moins tendance à devenir très grandes, ce qui ralentirait les opérations.

1.4.4 Couches de neurones

Comme mentionné précedemment, les neurones sont organisés en couches. Il y a 3 types de couches différentes. La première est la couche des intrants, dans laquelle les données sont rentrées dans le réseau. Dans le cas de notre programme, où les intrants sont des images de format 28x28, la première couche est composée de 784 ($28 \times 28 = 784$) neurones recevant chacun la valeur en échelle de gris d'un pixel de l'image. Plus concrètement, ces images sont des matrices carrées M_{28}, qui se font vectoriser en un vecteur de taille 784. Par la suite, chacune de ces données est transmise à chacun des neurones de la couche cachée, puisque le réseau est densément connecté, et les neurones d'une couche sont connectés à tout ceux des couches adjacentes. Pour la suite de cette explication, le réseau neuronal provenant de la figure affichée plus haut sera utilisé, à des fins de clarté. Donc, les valeurs des trois neurones de la couche d'intrants sont contenus dans la matrice $I_{1\times3}$. Les poids des neurones de la couche cachée 1 sont contenus dans la matrice $C_{4\times3}$, où 4 correspond au nombre de neurones dans la couche, et 3 aux poids que possèdent chaque neurones de la couche (un poid par neurone de la couche précédente). Ici, l'opération à faire serait un produit matriciel $A_{m\times p} \times B_{p\times n} = C_{m\times n}$, afin de multiplier les intrants par chaque ensemble de poids. Toutefois, les matrices ne sont pas compatibles pour effectuer cette opération, puisque le nombre de colonnes de la première matrice n'est pas égal au nombre de rangées de la seconde. Il faut donc faire la transposée de la matrice $C_{4\times3}$, qui devient alors $C^t_{3\times4}$. L'opération $I_{1\times3} \times C^t_{3\times4}$, où sont multipliés dans l'ordre, élément par élément, chaque élément d'une ligne de *I* par chaque élément d'une colonne de *C*, puis est effectué la somme de ces produits pour obtenir un nouvel élément de la matrice résultante $R_{1\times4}$ [Alloprof]. Par la suite, la matrice $B_{1\times4}$ contenant les biais de chaque neurone de la couche est additionée à la matrice R, dans une opération où s'additionnent entre-eux les éléments correspondants de chaque matrice pour former une nouvelle matrice de même dimension. Finalement, dans une itération au travers de cette matrice, chaque élément passe par la fonction d'activation, pour former encore une nouvelle matrice de même dimensions contenant les résultats de cette dernière opération. Cette matrice résultante finale $F_{1\times4}$ devient alors l'intrant de la couche suivante de neurones, et ainsi de suite.

1.4.5 Réseaux neuronaux et le cerveau humain

Plusieurs liens peuvent être faits entre les réseaux neuronaux et le cerveau humain. Le premier réseau neuronal était un système mécanique financé par la marine américaine qui tentait d'émuler les neurones biologiques. La fonction d'activation à échelons était utilisée, imitant les neurones biologiques qui s'activent *1* ou ne s'activent pas *0*. Le projet a rapidement été laissé de côté, principalement à cause du fait que le réseau était extrêmement difficile à entrainer, puisque, comme vu plus tôt, la fonction à échelon ne permet pas l'entrainement du réseau, et les paramètres devaient être ajustés au hasard.

Voilà donc une première différence fondamentale entre les neurones artificiels et organiques. Les neurones artificiels peuvent sortir toutes sortes de valeurs, de manière à mieux servir les intérêts du système, alors que dans le cas d'un neurone organique, elles ne peuvent envoyer que le signal binaire *activé* ou *non-activé*.

> Un neurone s'active lorsque son seuil d'excitation est atteint. Le potentiel de repos d'un neurone est d'environ -50mV. Lorsqu'il reçoit suffisament de neurotransmetteurs (des particules envoyées par d'autre neurones et qui possèdent une charge électrique), par ses dendrites et que le seuil d'excitation d'environ 15mV est atteint, le potentiel d'action se déclenche, et un influx nerveux se propage le long de l'axone sous forme de courant électrique. Une fois arrivé aux terminaisons axonales du neurone, d'autres neurotransmetteurs sont libérés par les synapses, poursuivant ainsi la transmission du signal. La quantité de neurotransmetteurs libérée ne dépend pas de l'intensité du stimulus initial ; c'est une situation de tout ou rien. [futura-sciences, n.d.]

En d'autres termes, l'image de la fonction d'un neurone artificiel *A*, dépendamment de la fonction d'activation, peut être, par exemple $\mathbb{R}$, $\mathbb{R}^+$, ou encore $[-1, 1]$, tandis que l'image de la fonction d'un neurone organique *O* est toujours limité à 0, 1.

L'aspect où les réseaux neuronaux et le cerveau humain ont le plus en commun est leur état initial. Les deux commencent comme un canvas vierge, ne possédant aucune connaissances ou expériences. Les deux se font « entrainer » par des informations extérieurs, jusqu'à arriver au point ou ils deviennent autonomes. Les connaissances qu'ils amassent se trouvent d'une certaine manière encodées dans leur système, et influencent leurs actions futures.

1.5 L'entrainement d'un système neuronal

L'entrainement d'un réseau à l'aide d'une certaine base de données (donnée d'entrainement) permet à celui-ci de prédire le résultat d'une autre base donnée. En effet, le but d'un réseau neuronal est de réduire l'erreur de l'entrainement ainsi que la différence entre l'erreur des données entrainées et l'erreur des données de test soient petites. Lorsque le réseau est sous-entrainé, le réseau de sera pas précis lors de ces résultats. Cependant, lorsque le réseau est sur-entrainé, celui-ci va prendre en compte tout le bruit des données. Ce bruit peut être, par exemple, le fait de prendre en compte les imperfections d'une image, reconnaitre seulement certains styles d'écriture, etc. Cela a comme impact d'augmenter l'erreur lorsque le système est exposé à une nouvelle base de données.

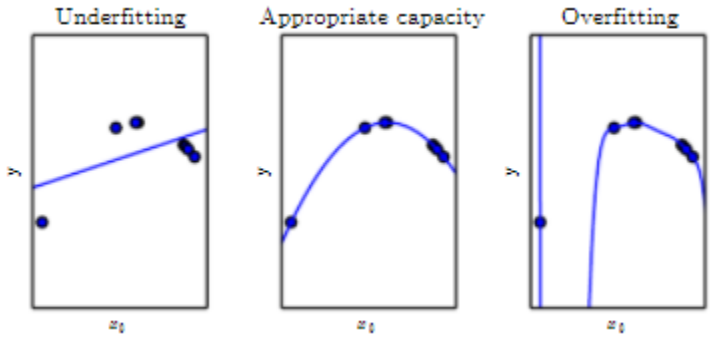


Fig. 1.5 – Graphiques représentant l'effet de l'entrainement du réseau de neurone

L'entrainement d'un réseau neuronal s'effectue à l'inverse. Visuellement, l'entrainement et l'ajustement des différents paramètres se font de la droite vers la gauche. Ce principe, appelé « backpropagation », va être expliqué à l'aide quelques démonstrations mathématiques complémentées par quelques explications écrites.

1.5.1 Fonction d'erreur

Une fonction d'erreur est une fonction permettant de connaitre la précision des résultats des extrants de la dernière couche. Il peut y avoir plusieurs fonctions d'erreur. En voici un exemple :

$E_{SS} = 1/2 \sum_{i=1}^{n} E_i^2$ **(1.1)**

$E_{SS} = 1/2 \sum_{i=1}^{n} E_i^2$ **(1.2)**

où E_{SS}= « error sum of square ». Cela est tout simplement une de plusieurs fonctions d'erreur.

$E_i = |t_i - I_i|$ **(1.3)**

où E_i correspond à l'erreur d'un neurone de la dernière couche (extrant). I_i correspond à la valeur numérique d'un extrant et t_i correspond à la valeur désirée provenant de la base de données fournies.

Combiner les deux équations permet d'obtenir :

$E = 1/2 \sum_{i=1}^{n} (T_i - Y_i)^2$ **(1.4)**

1.5.2 Transmition de l'information

> Note : Afin de simplifier les explications, ces dernières seront faites en utilisant un réseau neuronal ayant seulement 1 neurone par couche.

D'abord, il faut comprendre comment le réseau transmet son information de cellules en cellule. En effet, un neurone ayant contenant une certaine valeur Y transmet cette dernière à tous les autres neurones de la prochaine couche. Cependant, ces transmitions n'ont pas toutes les mêmes poids. Ces poids p diffèrent afin de favoriser certaines activations et en défavoriser d'autres. Chaque liaison entre chaque neurone possède un poid propre à chacune. Ces derniers sont multipliés avec l'extrant de la neurone en précédentes.

$Y_i = Y_{i-1} \times p_i$**(2.0)**

où p_i correspond au poid de la neurone de la couche i

Ensuite, un biais b est additionné ou soustrait au résultat précédent

$Y_i = Y_{i-1} \times p_i + b_i$ **(2.1)**

d'activation sera expliqué en détail plus loin.où b_i correspond au biais de la neurone de la couche i.

Finalement, une fonction d'activation a est ajoutée au reste de la formule. L'utilité et le fonctionnement de la fonction d'activation sera expliqué en détail plus loin.

$Y_i = a \times (Y_{i-1} \times p_i + b_i)$ **(2.2)**

1.5.3 *Back propagation*

L'objectif est de comprendre comment le poids et le biais doit être ajuster en débutant de la fonction d'erreur et d'activation.

Dabord, en utilisant la formule de base de transmission d'un neurone (sans le biais) :

$Y = \sum_{i=1}^{n} I_i \times p_i$

Il est possible de comprendre comment le changement d'une variable impact une autre. Les dérivés seront donc utilisées afin de démontrer ce principe.

$\frac{dY}{dI_i} = \frac{dY}{dI_i} \sum_{i=1}^{n} I_i \times p_{ji}$

$\frac{dY}{dI_i} = p_i$

$\frac{dY}{dp_i} = I_i$

Cela veut donc dire que le poid influence le résultat de l'extrant et que l'intrant influence le résultat de l'extrant.

En utlisant la formule (1.4) et le concept de dérivée partielle, il est possible de comprendre l'impact d'un changement de la valeur de l'intrant I_i sur l'erreur :

$\frac{dE}{dI_i} = (2/2)(t_i - I_i)(-1)$ $\frac{dE}{dI_i} = -(t_i - I_i)$

Maintenant, il faut calculer la dérivation de la fonction d'activation. La fonction sigmoïde sera utilisée pour cet exemple.

$a = \frac{1}{1+e^{-Y}} = (1 + e^{-Y})^{-1}$

$\frac{da}{dY} = -1(-e^{-Y})(1 + e^{-Y})^{-2}$

$= \frac{e^{-Y}}{(1+e^{-Y})^2}$

$= \frac{1}{(1+e^{-Y})} \times \frac{e^{-Y}}{(1+e^{-Y})}$

$= a \times \frac{e^{-Y}}{(1+e^{-Y})}$

$= a \times \frac{1+e^{-Y}-1}{(1+e^{-Y})}$

$= a \times (\frac{(1+e^{-Y})}{(1+e^{-Y})} + \frac{-1}{(1+e^{-Y})})$

$\frac{da}{dY} = a \times (1-a)$

Maintenant il est possible, à l'aide de la règle de dérivation en chaine, de trouver l'impact qu'a Y sur l'erreur E. Dans cet exemple, $I_i = a$ puisque la fonction d'activation été appliquée au neurone en question.

$\frac{dE}{dY_i} = \frac{dE}{dI_i} \times \frac{dI_i}{dY_i}$

$= \frac{dE}{dI_i} \times \frac{da}{dY_i}$

$= -(t_i - I_i)I_i(1-I_i)$ **(3.0)**

Ensuite il est possible de calculer la dérivation de l'erreur en fonction du poid p_{ji} d'une liaison entre deux neurones.

$\frac{dE}{dp_{ji}} = \frac{dE}{dY_i} \times \frac{dY_i}{dp_{ji}}$

$= (-(t_i - I_i) \times I_i \times (1-I_i)) \times I_i$

$= -I_i I_j (1-I_i)(t_i - I_i)$**(4.0)**

Cette équation signifie que le changement de l'erreur influence le poid et cette influence correspond à l'extrant d'un neurone négatif multiplié par l'extrant du neurone précédent et ce tout est multplié par 1 moins la valeur du neurone. À ce résultat est multiplié la valeur de l'erreur soit : $(t_i - I_i)$ L'équation 3.0 sera représenté par la variable : Δp

Le concept de « backpropagation » se résume donc a :

$p_{ji} = p_{ji} + \Delta p$

Le poids d'un neurone change légèrement en additionnant un Δp positif ou négatif. Ce changement est fait avec une plus grande importance plus le neurone est proche de la couche des extrants. Cela est dû au fait que l'apprentissage commence par la couche finale pour enfin se rendre jusqu'à la couche débutant le système neuronal. Les premiers ajustements, donc ceux des couches plus proches de la fin, sont plus importants. Les couches se situant plus au début du réseau vont plutôt avoir de petits changements à leur poids puisque rendu à l'ajustement de dce dernier, l'erreur est déja considérablement réduite. Ce concept se nomme descente de gradient stochastique.

###Origine du biais Certains problèmes peuvent survenir avec le concept de descente de gradiant dans un réseau neuronal. En effet, lorsqu'une couche « n'apprend plus » ou, en d'autres mots, lorsque le poids ne varie plus, on assiste a un problème se nommant la disparition du gradiant. Cela est un problème pour le réseau puisque le tout l'entrainement se fait uniquement dans les dernières couches. Un autre problème est que le gradiant dans une fonction de coût telle une sigoïde, le gradient se situe uniquement au milieu de la fonction comme le montre le graphique ci-dessous.

En effet, les extrémités de la fonction forment un plateau. Il n'y a donc pas de changement possible puisque soit l'intrant est multiplié par 1, ce qui ne change pas le résultat, ou bien soit le résultat est multiplié par 0 ce qui rend la valeurr nul et cela est néfaste pour un réseau neuronal. En effet, une valeur égale a 0 empêche l'entrainement du réseau puisque peut importe la variation du poid, $0 \times p$ sera toujours être égale à 0. C'est pour remédier à cette erreur qu'un biais est ajouté à la fonction.

$Y = \sum_{i=1}^{n} I_i \times p_i + b$

L'ajout de ce biais va permettre de conserver un apprentissage même lorsque la valeur d'un neurone est figée à 0.

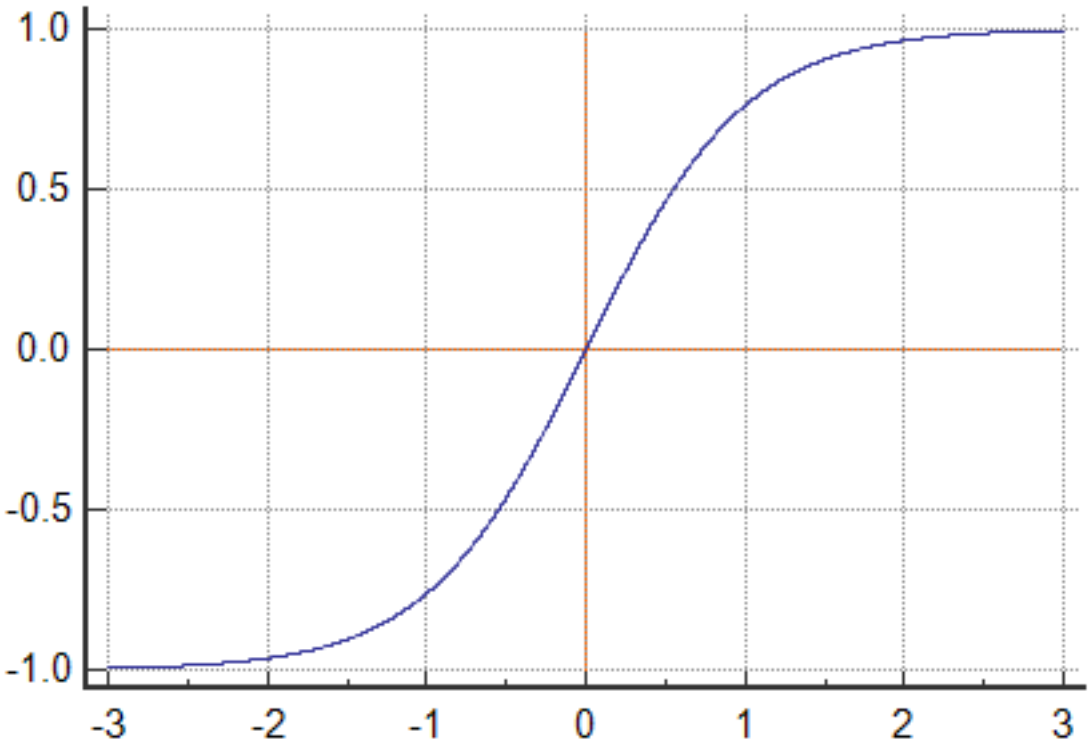


Fig. 1.6 – Fonction sigmoïde

1.6 Bienfaits et inconvénients

Il est important de parler des impacts positifs et négatifs de l'intelligence artificielle dans le but de bien comprendre ce que cette technologie peut apporter à l'évolution de la société. C'est pour cela que cette section va répondre à la sous-question de la thèse : Est- ce que l'intelligence artificielle peut constituer un bénéfice pour l'être humain.

Pour commencer, l'intégration de l'intelligence artificielle à notre vie de tous les jours ainsi que dans plusieurs tâches pourrait amener plusieurs bénéfices importants.

Qui dit programme, dit souvent un taux d'erreur qui diminue. En effet, l'erreur humaine est présente et joue un gros rôle dans plusieurs métiers. L'erreur est humaine , mais minimiser l'effort tout en maximisant la précision est primordial pour certaines tâches. L'utilisation d'une intelligence artificielle bien entraînée permet de réduire le taux d'erreurs et le temps requis à l'exécution d'une tâche. Cela est possible, car un programme assisté par l'intelligence artificielle peut parcourir un large jeu de données et appliquer des algorithmes en peu de temps, tandis que l'être humain prendrait des mois et des mois pour faire cela. Un exemple moderne serait l'utilisation de l'intelligence artificielle dans les prévisions météo. Ceux-ci deviennent donc de plus en plus précis avec l'aide de l'apprentissage machine ainsi que l'amélioration des algorithmes.

L'IA peut assister l'être humain dans des tâches dangereuses qui pourraient risquer la vie d'individus, mais qui est sans risques pour l'intelligence artificielle. Un exemple serait le robot Curiosity lancé en novembre 2011 dans le but d'atteindre la planète Mars. Il se trouve toujours sur cette planète en date du 19 novembre 2020 dans des conditions impossibles pour qu'un être humain y soit resté aussi longtemps. Celui-ci est contrôlé par l'intelligence artificielle dans le but d'accomplir plusieurs tâches. Ses tâches consistent à examiner la biologie, la géologie et la radiation de Mars, afin de préparer une exploration humaine dans le futur.

Les conditions de travail standard dictent qu'un être humain travaille environ 40 heures par semaine en incluant les pauses. De plus, ce ne sont pas toujours 40 heures productives. Les fins de journées et les fins de semaine sont souvent peu productives. En comparaison, un ordinateur n'a pas besoin de pause et peut travailler 24 heures sur 24, 7 jours sur 7. L'intelligence artificielle sera sûrement plus efficace qu'une majorité des travailleurs et aussi plus constante dans sa vitesse d'exécution. Cet ennui est souvent causé souvent par des travaux répétitifs. Ces tâches banales peuvent être automatisées par l'IA. Donnant plus de temps aux travailleurs pour se concentrer sur des tâches plus exigeantes et passionnantes. Comme les ordinateurs le font déjà dans plusieurs domaines, tel celui de la finance où un apprentissage machine s'occupe de détecter des fraudes potentielles dans les transactions des clients.

Quand on parle de rapidité, l'être humain est très lent comparé à l'intelligence artificielle. En effet, l'utilisation de l'intelligence artificielle pour la prise de décision permet de prendre ceux-ci extrêmement rapidement avec une précision qui dépend de la qualité de leurs bases de données. Effectivement, un jeu de données fiable et bien construit permet à l'intelligence artificielle de prendre des bonnes décisions rapidement. Un être humain, de l'autre côté, doit analyser une plus petite quantité de données directement sous son nez en prenant plus de temps. Un des exemples serait l'utilisation de l'intelligence artificielle dans les jeux d'échecs. L'intelligence artificielle à accès à des milliers et des milliers de parties dans son jeu de données. Alors, lorsque son adversaire humain joue le coup Nc3 (Cavalier sur la case C3), l'ordinateur analyse toutes les parties où ce coup c'est fait et en ressort le meilleur coup pour contrer celui-ci, et ce, en un temps record. De plus, la prise de décision de l'intelligence artificielle ne va pas seulement s'arrêter au jeu d'échec. Nous voyons déjà l'industrie médicale se faire aider dans la prise de décision par l'intelligence artificielle en plus des nouveaux programmes qui se font développer en ce moment. Ceux-ci ont pour but d'aider les professionnels de la santé dans des diagnostics, surveillances de patients, et plusieurs autres. La compagnie Google est en train de développer leur projet Google Health Pour se faire, ils étudient l'utilisation de l'intelligence artificielle dans le but d'assister au diagnostic de cancer, prévenir l'aveuglement et plusieurs autres.

La croissance exponentielle de l'intelligence artificielle dans le monde a affecté de manière abrupte le nombre de travail qui requiert des capacités en intelligence artificielle, et donc, a créé plusieurs nouveaux emplois dans ce domaine. Grâce à cela, la part des emplois qui requièrent des capacités en intelligence artificielle a augmenté de 4,5 fois depuis 2013 aux États-Unis sur le site web indeed.com.

Certes, ces merveilleux bénéfices ne viennent pas sans coût. Or, l'intelligence artificielle s'avère à être une arme à double tranchant et continents plusieurs problèmes

Un de ces problèmes majeurs est le biais de l'intelligence artificielle intégrée dans le modèle de manière involontaire ou intentionnelle par la façon dont celui-ci est entraîné ou programmé. Un exemple flagrant serait le programme Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) qui a signalé à tort des personnes ayant la peau foncée presque deux fois plus que des personnes ayant la peau blanche(45 % à 24 %) devant la cour juridique aux États-Unis. Un autre exemple serait l'intelligence artificielle programmée par Microsoft Corporation dans le but d'interagir avec les internautes. Prénommé «Tay», le bot s'est rapidement fait retirer après un entraînement avec l'interaction d'internautes racistes. Cela a mené le robot à dire des phrases vulgaires et inappropriées avant que celui-ci voit la fin de ses jours 16 heures après son arrivée sur Twitter. Le problème provient donc d'un jeu de données non fiable où mal construit qui induit l'apprentissage machine en erreur.

Tel que mentionné plus haut, l'intelligence artificielle a apporté et va apporter plusieurs nouveaux emplois spécialisés dans l'intelligence artificielle, mais plusieurs autres emplois vont voir leur fin arriver à grands pas avec la montée de l'intelligence artificielle. Cela c'est déjà fait avec la révolution industrielle, qui, avec l'arrivée de la machine à vapeur, avait fait grimper le taux de chômage à une vitesse fulgurante. L'intelligence artificielle devrait, tout comme la machine à vapeur, augmenter le besoin de certains emplois et créer des emplois qui requièrent un plus haut niveau d'étude en éducation tout en éradiquant des emplois auxquels aucune caractéristique uniquement humaine est requise. Il reste à noter que l'être humain s'en est quand même bien sortie de la révolution industrielle, contrairement aux idées pessimistes quant au futur proliférées à l'époque.

Par ailleurs, l'omniprésence de la technologie a créé une multitude d'interconnexions entre tous les pays. En plus, avec l'implémentation de l'intelligence artificielle, chaque pays va définir des lois et des règlements sur l'intelligence artificielle. Dans un monde idéal, tous les êtres humains s'entendent sur les mêmes lois et règlements pour éviter des conflits avec d'autres pays. Ces conflits seraient à cause de décisions par rapport à l'intelligence artificielle qu'un pays a choisi et qui mène à une contradiction avec un ou plusieurs autres pays. Malheureusement, comme l'Histoire nous l'a si bien démontré, essayer d'établir des règles communes s'avère difficile, et de les faire respecter, encore plus. Des différences se font déjà voir entre les puissances du monde face à l'intelligence artificielle avec l'Union Européenne qui est déjà entrain de pousser pour des mesures plus strictes pour le développement et l'utilisation de l'intelligence artificielle avec le 'White Paper on Artificial Intelligence – A European Approach to Excellence and Trust' ,publié en 2020. Au contraire, les États - Unis et la Chine permettent à leur compagnie d'utiliser l'intelligence artificielle plus librement.

L'intelligence artificielle va augmenter l'efficacité et la rapidité de plusieurs programmes. Cela n'exclut pas les programmes de piratages informatiques, qui eux aussi vont voir une amélioration drastique avec l'implémentation de l'intelligence artificielle. Cela va donc augmenter aussi la rapidité et l'efficacité des piratages informatiques menant sûrement à une augmentation de ceux-ci ce qui causera plusieurs problèmes majeurs, jusqu'à temps qu'une solution soit trouvée.

L'utilisation de l'intelligence artificielle dans des buts de tuer des individus est un grave danger. Tel que mentionné dans le rapport, l'entraînement de l'intelligence se fait à partir d'un grand jeu de données fiables. Dans un contexte d'une guerre contre le terrorisme, avoir un jeu de données sur les terroristres s'avère très difficile et peu fiable, car la plupart d'entre eux s'habille comme des civiles. De plus, cette technologie, une fois tombée dans les mains des terroristes, pourrait semer terreur au sein d'un pays comme le démontre très bien la vidéo Slaughterbots qui promeut l'interdiction de l'usage robot tueur.

En conclusion, l'implémentation de l'intelligence artificielle ainsi que la croissance exponentielle de celle-ci va rapporter une tonne de bénéfices tel l'élimination de l'erreur humaine, l'utilisation de celle-ci dans des tâches dangereuses, la disponibilité 24/7 de celle-ci, la rapidité de prise de décision et les emplois dans ce domaine. De l'autre côté, si l'on veut que ces bénéfices portent fruit, il faut limiter ou éliminer les impacts négatifs de l'intelligence artificielle. Cela va se faire graduellement par l'implémentation de règles mondiales sur les jeux de données utilisés dans l'intelligence machine, l'intelligence machine en temps que tel, l'utilisation de l'intelligence artificielle dans un environnement de guerre, et ce, dans l'accord de la majorité des pays en plus de développer et améliorer la cybersécurité. Une idée intéressante pour régler le problème des jeux de données pourrait être de standardiser les jeux de données qui ne sont pas biaisés dans le but d'éviter les problèmes. C'est déjà ce que le NIST fait en fournissant des jeux de données fiables gratuitement.

Finalement, quel est le fonctionnement de l'intelligence artificielle et comment devrait-elle être utilisée afin de bénéficier l'être humain ? L'hypothèse émise était que si son développement se fait de manière éthique et s'il est bien encadré,

nous pourrions en retirer plus d'avantages que d'inconvénients. Il était donc important de comprendre le fonctionnement de cette technologie pour pouvoir expliquer et rationaliser l'utilisation bénéfique de l'intelligence artificielle et d'ainsi le documenter.

1.7 Conclusion

Finalement, quel est le fonctionnement de l'intelligence artificielle et comment devrait-elle être utilisée afin de bénéficier l'être humain ? L'hypothèse émise était que si son développement se fait de manière éthique et s'il est bien encadré, nous pourrions en retirer plus d'avantages que d'inconvénients. Il était donc important de comprendre le fonctionnement de cette technologie pour pouvoir expliquer et rationaliser l'utilisation bénéfique de l'intelligence artificielle et d'ainsi le documenter.

Pour ce faire, il a fallu procéder à l'écriture d'un programme d'OCR, une forme simple d'intelligence artificielle qui a pour but de reconnaître des caractères écrits à la main. Grâce à l'information acquise lors de l'écriture de celui-ci ainsi que toute la documentation lue pour la préparation du programme, il a été plus facile de comprendre les aspects d'un réseau neuronal dans le but de documenter chaque composante du réseau neuronal.

Ce que nous pouvons conclure d'un réseau neuronal après notre documentation, c'est que cette "intelligence" artificielle, n'est rien d'autre qu'un paquet de fonctions avec des paramètres qui ont été ajustés par une méthode d'optimisation dénommée "Back propagation". Cette méthode est composée de fonctions qui utilisent des notions de mathématiques telles que les dérivées. Cet algorithme a lieu lors de chaque "epoch [2]" dans le but d'ajuster l'algorithme d'OCR graduellement.

Donc, cet algorithme « intelligent " n'est pas capable de prendre des décisions elle-même et doit être surveillé et entraîné par un être humain à l'aide d'un jeu de données.Cela dans le but de pouvoir répondre à la tâche précise à la-quel l'algorithme s'est fait assigner, qui est dans notre cas la reconnaissance optique des chiffres de 0 à 9. Alors, un algorithme qui est bon ou mauvais dans sa tâche et donc qui a une bonne ou mauvaise précision est principalement déterminé par la façon dont l'algorithme a été entraîné et si le jeu de données utilisé pour l'entraîner est fiable et diversifié. Dans notre cas, la précision est calculée par le nombre de prédiction réussite sur le nombre total de prédiction.C'est donc l'entraînement que provient une partie des biais de l'intelligence artificielle, et donc un des inconvénients qu'on avait cité dans notre hypothèse.

L'autre inconvénient qui a été mentionné lors de l'hypothèse est l'importante quantité d'emploi qui risque de disparaître. Or, tel que mentionné dans la section sur les impacts de l'intelligence artificielle, il n'y a pas seulement cet inconvénient, et ces autres inconvénients sont accompagnés de plusieurs avantages intéressants.

Comme il est décrit dans la section des bienfaits et inconvénients, l'usage de l'apprentissage machine peut amener beaucoup d'avantages ainsi que des inconvénients. Ces principaux avantages se résument à, l'élimination de l'erreur humaine, l'utilisation de l'IA dans des tâches dangereuses, la disponibilité 24/7 de celle-ci, la rapidité de prise de décision et les emplois dans ce domaine. Tandis que les inconvénient se résument au biais de l'IA, diminution de certains types d'emploi, les conflits mondiaux lié à celle-ci, l'amélioration des "hacks" et l'utilisation de celle-ci dans le but de tuer. Si l'on veut avoir les bénéfices de l'intelligence artificielle, il faut d'abord s'occuper de diminuer l'impact des inconvénients. La bonne nouvelle étant que ces inconvénients peuvent être quasi inexistants si l'IA est implémentée en suivant dans un bon cadre strict ainsi que des mesures de sécurité. Par exemple, des règles sur l'utilisation des bases de données et des réglementations sur celles-ci pourraient diminuer les biais dans les programmes d'apprentissage machine. Si ces conditions sont respectées, il est préférable de penser que l'usage de l'intelligence artificielle pourrait faire évoluer la société.

Pour répondre à la thèse, l'intelligence artificielle fonctionne par l'étude de grosses bases de données par un programme. Celui-ci va ensuite faire varier ses paramètres clés dans le but d'ajuster certaines fonctions. Ces fonctions permettent de trouver des tendances dans le jeu de données et de les utiliser dans le but de répondre à la tâche attribuée. De plus, l'intelligence artificielle peut constituer un bénéfice pour l'être humain à condition que celle-ci soit bien encadrée.

2. Une "epoch" étant un cycle complet où l'algorithme a traité le jeu de données qui lui a été fournie une seule fois.

Il ne faut toutefois pas écarter le fait que le futur n'est jamais certain, et qu'on ne peut prédire à 100% ce que l'intelligence artificielle va ressembler dans 50 ans. Mais, avec les avancées technologiques des dernières années qui ne semblent pas s'arrêter, il serait juste de croire que l'IA a un futur prometteur. Avec ces avancés prometteuses, ce pourrait-il qu'il y ait une sorte d'intelligence au-delà de celle humaine et artificielle dans le futur ?

1.8 Bibliographie

Bibliographie

[InnovationQuebec, 2018] et Innovation Québec, É. (2018). *Les avantages et inconvénients de l'intelligence artificielle*.

[futura-sciences, n.d.] futura-sciences (n.d.). *Potentiel d'action*.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http ://www.deeplearningbook.org.

[Grother et al., 2019] Grother, P., Ngan, M., & Hanaoka, K. (2019). *Face Recognition Vendor Test (FRVT) Part 3 : Demographic Effects*. NIST.

[LAURO, n.d.] LAURO, D. M. (n.d.). *HUMAN BRAIN AND NEURAL NETWORK BEHAVIOR A COMPARISON*.

[Metz, 2020] Metz, R. (2020). *Portland passes broadest facial recognition ban in the US*.

[Nielsen, 2019] Nielsen, M. A. (2019). *Neural Networl and Deep Learning*.

[Popper, 2016] Popper, N. (2016). *The Robots Are Coming for Wall Street*.

[RFisher & Wolfart, 2003] R. Fisher, S. Perkins, A. W., & Wolfart, E. (2003). *Pixel Values*.

[Les contributeurs de Wikipedia, 2020] Les contributeurs de Wikipedia (2020). *TensorFlow*.

[The Numpy documentation team, 2020] The Numpy documentation team (2020). *About us*.

[The TensorFlow developper team, 2015] The TensorFlow developper team (2015). *TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems*.

[The TensorFlow developper team, 2020] The TensorFlow developper team (2020). *TensorFlow*.