

1 Les premières intelligences artificielles

1.1 Introduction et définition

Nous pouvons tout d'abord définir l'intelligence artificielle. L'intelligence artificielle (IA, ou AI en anglais pour Artificial Intelligence) a pour but de mimer ou d'imiter une certaine forme d'intelligence réelle. Elle essaie de se substituer à la logique de la prise de décision d'un humain en mettant en œuvre différentes techniques.

L'intelligence artificielle est définie comme « l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence ». Elle correspond donc à un ensemble de concepts et de technologies plus qu'à une discipline autonome constituée.

La notion d'intelligence artificielle est née dans les années 50 et a été mise en avant par deux chercheurs. Tout d'abord, le mathématicien Alan Turing a débuté ses recherches dans ce domaine car il se demandait si une machine pouvait penser. Dans son article Computing Machinery and Intelligence , il cherche à savoir s'il est possible d'apporter aux machines une forme d'intelligence. Turing a analysé le problème et a proposé une expérience connue aujourd'hui sous le nom de « Test de Turing ». Dans ce test, un sujet interagit à l'aveugle avec un autre humain, puis avec une machine programmée pour formuler des réponses sensées. Si le sujet n'est pas capable de faire la différence, alors la machine a réussi le test et, selon l'auteur, peut véritablement être considérée comme « intelligente ». Ce test permet donc de définir si une machine est consciente. Ensuite, Warren Weaver a publié en 1949 un mémo-randum. Celui-ci se base sur la traduction automatique des langues, ce qui signifie qu'une machine est capable de faire des tâches dites humaines. Il est principalement connu comme un des pionniers de la traduction automatique

L'intelligence artificielle en tant que domaine de recherche a été créée lors d'une conférence qui se déroulait sur le campus de Dartmouth College pendant l'été 1956.

Les organisateurs de la conférence de Dartmouth prévoient que soient abordés diverses questions autour de l'idée de machine pensante : Comment simuler la pensée et le langage au travers de règles formelles ? Comment faire penser un réseau de neurones ? Comment doter une machine de capacité d'apprentissage automatique ? Comment doter une machine de créativité ? Néanmoins, les discussions seront assez limitées, et la conférence de Dartmouth n'aura servi qu'à forger un embryon de communauté de recherche

autour des problématiques évoquées. McCarthy propose le terme d'intelligence artificielle pour désigner la nouvelle discipline qui va connaître son âge d'or durant les 15 années qui suivent.

Elle s'est ensuite fortement développée aux États-Unis grâce à certaines personnes de renom dont John McCarthy à l'université Stanford, Marvin Minsky au MIT, Allen Newell et Hebert Simon à l'université de Carnegie-Mellon et pour finir de Donald Michie à l'université d'Edimbourg. La plupart d'entre eux ont reçu des prix Turing : Marvin Minsky en 1969 ; John McCarthy en 1971 ; Allen Newell et Herbert Simon en 1975. D'autres piliers de l'intelligence artificielle ont reçu ces prix : Edward Feigenbaum et Raj Reddy en 1994 ainsi que Judea Pearl en 2011.

Nous détaillerons tout cela plus loin.

1.2 Quelques grands précurseurs de l'IA

1.2.1 Des scientifiques et des ingénieurs

Konrad Zuse (né le 22/06/1910 – Décédé le 18/12/1995) est un ingénieur allemand et l'un des fondateurs du calcul programmable. Entre 1936 et 1938, Konrad Zuse développe Z1, le premier calculateur mécanique fonctionnant un moteur électrique. En parallèle de la conception de Z1, en 1937, l'ingénieur allemand crée le premier calculateur électro-mécanique programmable binaire à virgule flottante, le Z3. Il est opérationnel en 1941 soit 4 ans de travail.

Claude Shannon (né le 30/04/1916 – Mort le 24/02/2002) est un ingénieur électricien et un mathématicien américain. Il est l'un des pères fondateurs de la « Théorie de l'information ». Il donne son nom à un schéma qui prend le nom de « Schéma de Shannon ». Cet outil est utilisé en Sciences Humaines et dans la Communication. Lorsqu'il était mathématicien, il utilise l'Algèbre de Boole dans le but de mettre en place des circuits de commutation. Il apporte sa pierre à l'édifice en mettant en place un outil théorique aux concepteurs de circuits logiques. Il se tourne vers l'informatique. Il commence à créer une machine à jongler, une souris parcourant les labyrinthes, une machine qui résout le cube de Rubik, une autre machine permettant de jouer aux échecs.

Allen Newell (né le 19/03/192 - mort en 19/07/1992) a été chercheur en informatique ainsi qu'en psychologie cognitive au sein de deux organismes : RAND Corporation et à la Carnegie-Mellon's School of Computer Science

aux Etats Unis. Il participe à l'élaboration de différents programmes : "Information Processing Language" en 1956 et à deux programmes en Intelligence Artificielle "The Logic Theory Machine" (1956) et le "General Problem Solver" (1957). Pour le dernier programme, il travaille en collaboration avec Herve Simon.

John Mc Marthy (né le 04/09/1927 - Mort le 24/10/2011) est l'un des principaux pionniers de l'Intelligence Artificielle avec Marvin Minsky. Il met l'accent sur la logique symbolique. En 1948 John McCarthy obtient un Bachelor of Science en mathématiques au California Institute of Technology, puis un doctorat à Princeton en 1951. Sa thèse porte sur un certain type d'équations aux dérivées partielles, mais son passage à Princeton lui fait rencontrer Minsky avec qui il se découvre une passion commune pour l'idée de machine pensante. A l'âge de 28ans, il élabore un algorithme permettant d'évaluer et jouant un rôle majeur dans la programmation en Intelligence Arficielle. Cette algorithme est utilisé dans le plus part des programmes d'échecs. En 1958, il met au point le langage LISP. Il crée à l'Université Standord le laboratoire de l'Intelligence Artificielle.

1.2.2 Un économiste

Herbert Simon (né le 15/06/1916 - mort le 09/02/2001) était avant tout un économiste et un sociologue américain. Il a utilisé pour la première fois les ordinateurs et a conclu que l'ordinateur avait deux rôles à savoir la reproduction de la pensée humaine et la systématisation la pensée humaine. Sa rencontre avec Allen Newell a été primordiale car il s'est penché sur les activités intellectuelles humaines et a découvert qu'elles pouvaient etre automatisées. Ils ont conçu le "General Program Solver" en 1957. Ils ont développé l'Intelligence Artificielle de différentes manières.

1.2.3 Les influences littéraires

Isaac Asimov (né le 02/01/1920 en Russie - Mort le 06/04/1992 aux Etats Unis) est un écrivain américain. Il a écrit des oeuvres de science-fiction ainsi que les livres de vulgurisation scientifique. Il a écrit une série d'histoires (série des Robots) sur les rapports conflictuels entre l'homme et la machine et le role de robots.

Hubert Dreyfus (né le 15/10/1929) est un professeur américain de philosophie à l'université de Californie. Il s'intéresse à différents sujets comme le

phénoménologie, l'existentialisme, la philosophie de la psychologie, la littérature et bien sûr l'Intelligence Artificielle. Il critique fortement Allen Newell et Herbert Simon dans le livre "Alchemy and Artificial Intelligence"

1.3 Apparition des premiers ordinateurs

Les années 1940 et 1950 voient l'apparition des premiers véritables ordinateurs. Ils sont Turing complets et électroniques, donc (relativement) rapides. Les entrées et sorties se font par cartes perforées et impression papier. La programmation de telles machines est compliquée et très longue. Il n'existe que très peu d'ordinateurs, uniquement dans quelques universités ou grandes entreprises. Ces machines couteuses servent surtout à faire des calculs massifs(statistiques pour l'état, calculs scientifique pour la recherche nucléaire, calculs balistiques pour l'armée, ...). Par exemple, l'UNIVAC I (Universel Automatic Computer) est installé en 1951 au bureau du recensement américain.

Notons bien que l'informatique n'existe pas encore. Les spécialistes des ordinateurs sont en effet essentiellement des mathématiciens ou des électro-niciens. Les langages de programmation au sens moderne du terme n'existent pas encore : le premier langage évolué, FORTRAN (FORmula TRANslator) ne verra le jour qu'en 1954. Neanmois l'apparition des ordinateurs semble rendre possible le rêve de l'IA. Les deux approches de l'IA vont émerger dans les années 1940 à savoir le connexionnisme et le cognitivisme.

Le connexionnisme modélise les phénomènes mentaux ou comportementaux comme des processus émergents de réseaux d'unités simples interconnectées. Le plus souvent les connexionnistes modélisent ces phénomènes à l'aide de réseaux de neurones.

Le cognitivisme, lui, est le courant de recherche scientifique endossant l'hypothèse selon laquelle la pensée est analogue à un processus de traitement de l'information. Il considère que la pensée peut être décrite à un niveau abstrait comme manipulation de symboles, indépendamment du support matériel de cette manipulation (cerveau, machine électronique, etc). Elle est définie en lien avec l'intelligence artificielle comme une manipulation de symboles ou de représentations symboliques effectuée selon un ensemble de règles. Cette approche établit un lien entre la pensée et le langage (système de symboles).

Le 20eme siècle voit en effet apparaître plusieurs théories comme la cybernétique et le cognitivisme pour modéliser l'esprit, le cerveau et le mode de fonctionnement de la pensée.

De surcroît, dans le contexte de la guerre froide, la traduction automatique du russe en anglais ou de l'anglais au russe paraît cruciale. En 1954, un premier programme, écrit à l'université de Georgetown permet de traduire plusieurs dizaines de phrases simples. Le programme utilise 250 mots et seulement 6 règles de grammaire et tourne sur un IBM 701.

Des crédits sont rapidement alloués aux recherches sur la traduction automatique (aussi bien aux USA qu'en URSS). Les premiers travaux visent la traduction directe, presque mot à mot, à l'aide de dictionnaires bilingues et de règles simples. Les problèmes de polysémie (amateur, blanc) ou d'homonymie (mousse, avocat, ...) apparaissent très rapidement.

1.3.1 Les travaux de Simon et Newell

Allen Newell (1927-1992), après un Bachelor of Science en physique à Stanford, rejoint Princeton en 1949 pour mener une thèse en mathématiques. Durant ses études, il a été fortement influencé par le mathématicien hongrois Georges Polya (1887-1985), qui avait introduit la notion d'heuristique pour la résolution de problème. Une heuristique (du grec eurisko, je trouve) est une méthode empirique de résolution de problème, dont la validité ou l'efficacité n'est pas prouvée. Par exemple, protéger la reine aux échecs, choisir la caisse où la file est la plus courte, ... Trouvant finalement les mathématiques trop abstraites, Newell accepte en 1950 un poste à la RAND Corporation de Santa Monica, pour mener des travaux plus concrets, sur l'aéronautique de défense notamment. Simon est également consultant à la RAND (Research ANd Development). La RAND, créée pour étudier la mise au point d'un satellite artificiel, va peu à peu étendre ses travaux à l'informatique, l'économie, la géopolitique, etc. Les idées de Simon et de Newell convergent : La rationalité limitée de Simon implique que la prise de décision repose sur des procédures permettant de palier aux manques d'information en tenant compte du contexte. Pour Newell, ces procédures sont des heuristiques.

Simon et Newell considèrent que la condition nécessaire et suffisante pour qu'une machine puisse faire preuve d'intelligence est qu'elle soit un système physique de symboles.

Mais ils mettent au cœur de leurs travaux la notion d'heuristique : être intelligent, c'est aussi être capable de construire des heuristiques, de les tester, de les faire évoluer. Aidés par un programmeur de la RAND, Cliff Shaw, ils développent Logic Theorist en 1956, un programme de démonstration automatique de théorème (voir l'article The logic theory machine : A Complex

Information Processing System, 1956). Logic Theorist est considéré comme le premier programme informatique relevant du domaine de l'IA.

Newell et Simon y démontrent une série de théorèmes et envoient un article avec cette nouvelle démonstration au Journal of Symbolic Logic, en ajoutant Logic Theorist comme co-auteur. Mais l'article est refusé au motif que ce théorème est déjà démontré depuis longtemps. Simon écrit dans son autobiographie : « nous avons inventé un programme informatique capable de penser de façon non-numérique et, de ce fait, avons résolu le vénérable problème de l'âme et du corps, en expliquant comment un système composé de matière pouvait exhiber les propriétés de l'esprit » (Models of My Life, H. Simon, 1991).

Les travaux de Newell et Simon illustrent également les apports croisés entre l'informatique et l'intelligence artificielle. On a d'une part le développement de l'informatique rend possible des expériences en IA. Et d'autre part les problèmes posés par les expériences en IA conduisent à produire des outils qui servent le développement de l'informatique. Pour faciliter la programmation du Logic Theorist, Newell, Simon et Shaw développent le langage IPL (Information Processing Language) en 1956.

1.4 Le Japon et l'IA

Bénéficiant des techniques de l'intelligence artificielle, la transmission des savoirs vit au début des années 90 une révolution. Les concepts sont bouleversés. Ils s'enrichissent de l'apport du couple IA-Robotique qui crée ses propres spécificités.

Si les robots font déjà partie intégrante du paysage industriel japonais, où ils servent à réaliser des tâches dangereuses ou nécessitant une haute précision de manière automatique, une évolution majeure du marché se dessine avec l'émergence de la robotique de service et, dans une moindre mesure, de la robotique de loisir.

Les Japonais sont depuis très longtemps les champions de la robotique et on ne compte plus leurs inventions. Les recherches actuelles concernent tout particulièrement les robots « humanoïdes », qui tentent d'imiter les humains et leurs capacités.

Par exemple les robots humanoïdes présentent plus de difficultés de développement, puisqu'il s'agit d'imiter des facultés humaines, ce qui n'est pas simple. Le Japon rencontre un problème très particulier avec une natalité très faible et donc un vieillissement très important de la population qui nécessite

de plus en plus d'aide. C'est pour cette raison que Honda continue de travailler sur ASIMO, le robot humanoïde qui devrait aider les personnes âgées à rester à leur domicile, en les assistant dans de nombreuses tâches qu'ils ne peuvent assumer. Par ailleurs, des chercheurs de l'université de Tohoku, associés à des chercheurs français, consacrent leurs travaux à la résolution d'un problème très important : permettre à un robot humanoïde, HRP-2, de se déplacer sur des surfaces irrégulières. En effet, ce type de robot ne peut se déplacer que sur une surface dure et stable, ce qui limite fortement son champs de manœuvre. La résolution de ce problème représenterait une avancée considérable en robotique et donc également en intelligence artificielle.

1.5 Aujourd'hui

D'années en années, l'IA a été implémentée dans de plus en plus de domaines d'application et le développement de celle-ci est au centre de beaucoup de recherches actuelles. Le développement des technologies informatiques et des techniques algorithmiques ont eu une telle croissance ces dernières années que les machines dépassent l'homme dans plusieurs domaine comme le jeu d'échecs ou encore le jeu de go.

Toutefois, les sujets concernés par l'IA ont varier au cours du temps. Par exemple, dans les années 1950, la recherche d'un itinéraire était considéré comme un problème d'intelligence artificielle, alors qu'aujourd'hui, il existe différentes applications qui sur base d'algorithmes résolvent ces questions de recherche d'itinéraires et il ne s'agit plus d'intelligence artificielle.

Aujourd'hui toutes les grandes entreprises dans le monde de l'informatique (Google, Microsoft, Apple, IBM ou Facebook) portent une attention toute particulière aux problématiques de l'intelligence artificielle en tentant de l'appliquer à différents domaines précis. Chacun met ainsi en place des réseaux de neurones artificiels constitués de serveurs afin de traiter de lourds calculs au sein de gigantesques bases de données.

Aujourd'hui, l'intelligence artificielle est de plus en plus utilisée pour être au service des humains. Ainsi, par exemple, la vision artificielle comme nous l'expliquerons dans ce travail, permet à la machine de déterminer précisément le contenu d'une image comme des pièces de monnaies pour ensuite la classer automatiquement selon l'objet, la couleur ou le visage repéré.

La reconnaissance vocale a également le vent en poupe avec des assistants virtuels capables de transcrire les propos formulés en langage naturel puis de traiter les requêtes soit en répondant directement via une synthèse vocale,

soit avec une traduction instantanée ou encore en effectuant une requête relative à la commande. Apple était en la matière un des précurseurs avec son application « SIRI »

Nous pouvons donc affirmer que l'IA qui était au départ très restreinte a un potentiel infini. Sans cesse, de jour en jour, les recherches progressent et les résultats de ce que l'intelligence artificielle est capable de réaliser croît de manière exponentielle.

2 Techniques de reconnaissance de pièces

2.1 Machine learning

Le machine learning ou en bon français apprentissage automatique est l'élaboration d'algorithmes capable d'apprendre en étudiant des exemples. Il est la branche principale de l'intelligence artificielle. Le machine learning est particulièrement bien adapté à la reconnaissance d'image et ainsi à la détection de pièces de monnaie.

2.1.1 Principe général

Le machine learning est constitué de deux étapes principales.

1. Phase d'apprentissage : l'algorithme traite des milliers d'information d'une base de donnée. Il reçoit une entrée X dont il connaît la sortie Y . Lors de cette phase, l'algorithme va analyser les corrélations qui lient les entrées X aux sorties Y . Il va en déduire de lui-même les caractéristiques importantes de cette liaison.
2. Phase de prédiction : l'algorithme reçoit une nouvelle entrée X , il la compare à celles qui l'a analysées et ainsi, il doit prédire la sortie Y .

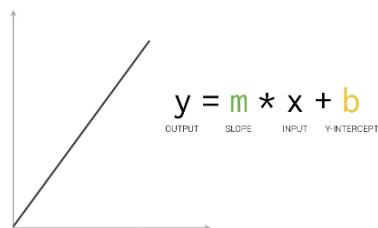
Dans notre cas, l'entrée X est l'image d'une pièce de monnaie et la sortie Y la valeur de la pièce. L'algorithme va s'entrainer à reconnaître une pièce de monnaie et à les distinguer les unes des autres en analysant des milliers d'images d'une base de donnée. Ainsi, il sera capable face à une nouvelle image d'en déduire la valeur de la pièce.

2.1.2 Application détaillée du principe

Le machine learning peut être détaillé en 7 étapes et ainsi être mieux expliqué.

1. Rassemblement de données : actuellement, il existe des milliers de base de donnée regroupant chacune d'elle des milliers d'images. Cette première étape consiste à rassembler les images qui nous intéressent.
2. Préparation des données : c'est l'encodage et la détermination de l'entrée X et de la sortie Y. On doit imaginer pouvoir interpréter les données par un repère et des axes X et Y.
3. Choix du modèle : le modèle est un algorithme mathématique avec un certain nombre de paramètres qui doivent être appris à partir de données. La partie essentielle du machine Learning est de trouver un modèle qui puisse correspondre au mieux à ces données.
4. Entrainement : le but de l'entraînement est de créer un modèle précis qui puisse répondre à notre question de façon fiable. On a besoin de données pour entraîner le module. On va prendre des caractéristiques de l'objet que l'on veut reconnaître (taille, forme couleur,...).

Training



On place les paramètres sur droite $d \equiv y = mx + b$ où

$$\begin{cases} x \text{ est l'entrée} \\ m \text{ est la pente} \\ b \text{ est l'ordonnée à l'origine} \\ y \text{ est la valeur de } d \text{ en un point précis, c'est la sortie} \end{cases}$$

Pour modifier la position de la droite, on joue sur les paramètres m et b (m et b sont les différentes valeurs des caractéristiques). Pour que le système puisse assimiler toute l'information, on utilise des matrices. La matrice des m est la matrice des poids (*weights*) et la matrice des b est la matrice des biais (*biases*).

Training

$$\begin{aligned}
 \text{WEIGHTS} &= \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \\ m_{3,1} & m_{3,2} \end{bmatrix} \\
 \text{BIASES} &= \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}
 \end{aligned}$$

Il faut un système avec tous les paramètres m et tous les paramètres b et le but est de, en faisant varier les paramètres, entraîner le système à reconnaître et à faire des prédictions de plus en plus fiable.

5. Evaluation : après l'entraînement, une évaluation du système avec des nouvelles données nous aide à voir le pourcentage de fiabilité du module.
6. Paramètre et hyperparamètre : ce sont deux types de paramètre. Ces paramètres expriment les propriétés de «niveau supérieur» du modèle telles que la complexité et la rapidité d'apprentissage. Les hyperparamètres, ne pouvant pas être directement appris sur base de l'entraînement, sont généralement fixés avant-même que le processus d'apprentissage commence. De manière générale, on fixe la valeur de l'hyperparamètre, après avoir testé différentes valeurs de celui-ci et en décidant lesquels fonctionnent le mieux.
7. Prédiction : il ne reste plus qu'à faire tourner le système avec les données qui nous intéressent pour obtenir la reconnaissance des différentes pièces monnaies.

2.1.3 Différents type de machine learning

2.1.3.1 Apprentissage supervisé

L'apprentissage supervisé est la forme de machine learning la plus répandue. Elle consiste à entraîner l'algorithme sur des entrées X dont on connaît les sorties Y . Il y a eu donc au préalable une intervention d'un expert qui a lié les entrées X et les sorties Y entre elles.

2.1.3.2 Apprentissage non-supervisé

L'apprentissage non-supervisé consiste à faire aucun lien entre les entrées et sorties. L'algorithme doit trouver par lui-même la structure cachée entre les données. Ainsi, aucun expert n'est requis. Et la phase de prédiction est dès lors impossible : l'algorithme classe uniquement les données.

2.1.3.3 Apprentissage par renforcement

Par ce dernier apprentissage, l'algorithme modifie son comportement au vu des observations et des résultats qu'il tire de ses expériences. Chaque nouvelle action de l'algorithme lui permet de connaître mieux son environnement et ainsi d'optimiser davantage son apprentissage.

2.1.4 Le deep learning

2.1.4.1 Limitations du machine learning conventionnel

Dans des problèmes plus compliqués, on peut avoir plus qu'une seule donnée en entrée x , ce qui donne une relation graphique de plus en plus complexe au plus on ajoute des entrées. On se rend rapidement compte que la simple droite ne pourra pas répondre à nos attentes en reconnaissance d'image dans beaucoup de situations. Une image est composée de millions de pixels et en machine learning il devient rapidement compliqué de gérer une telle quantité de données en entrée. C'est ici qu'intervient le deep learning.

2.1.4.2 Des neurones en réseau

Le deep learning est une façon de faire du machine learning. Il utilise des neurones artificiels pour trouver la sortie y . Un neurone (artificiel) est une construction mathématique qui va mettre en relation des entrées X avec une sortie Y .

Exemple On pourrait par exemple avoir un neurone qui, pour une série de n entrées x_i et de n poids w_i (avec $i \in [1, n]$), calculerait la somme $s = \sum_{i=1}^n x_i \cdot w_i$. Selon que cette somme serait $\geq S$ ou $< S$ où S est un seuil fixé, le neurone renverrait 1 ou 0 respectivement. On peut ainsi créer un système binaire.

Ce n'est bien sûr qu'un exemple, mais il montre bien la polyvalence d'un neurone. Dans des situations plus complexes, on peut combiner les neurones et créer alors un réseau de neurones. Pour augmenter la fiabilité des résultats, il faut passer par une phase d'entraînement avant d'entrer dans la phase de prédiction, comme dans le machine learning. Pour adapter au mieux la justesse des réponses, l'algorithme joue sur les poids et les seuils.

Un des gros inconvénient du deep learning est qu'au plus on ajoute de couches de neurones, au plus la phase d'entraînement est longue et au plus il devient difficile de trouver les bonnes valeurs de poids et seuil.

2.1.4.3 Application à la reconnaissance d'images

Sur une image brute, il y a beaucoup de pixels. Il est donc très difficile de pouvoir établir un réseau de neurones. Pour pouvoir reconnaître des images, il est plus simple de faire abstraction de l'image et de créer un algorithme intermédiaire qui aura pour but d'analyser toutes les caractéristiques intéressantes. Ce sont ensuite ces caractéristiques qui seront données à un réseau de neurones simple pour faire la reconnaissance. Cette approche est plus efficace. L'algorithme filtre déjà une partie de l'information intéressante pour la reconnaissance. Le seul problème de cette méthode est que la qualité de la reconnaissance va fortement dépendre du choix des caractéristiques par l'algorithme et du réseau de neurones.

Cependant, on peut faire un réseau profond, c'est un très gros réseau de neurones auxquels on peut directement donner l'image brute. Si l'on parvient à correctement entraîner ce réseau, on se rend compte qu'après l'entraînement, le réseau a lui-même découvert les caractéristiques à sélectionner dans les images. Il y a une architecture particulière à respecter pour que le réseau soit le plus efficace possible. Cette méthode n'a pu se développer fortement ces dernières années que grâce à la grande disponibilité des données. A l'heure actuelle, on peut faire un réseau de deep learning de plus d'une centaine de

couches constituées de plusieurs millions de neurones. Les performances en reconnaissance d'images sont assez spectaculaires.

2.2 Transformée de Hough

2.2.1 Introduction

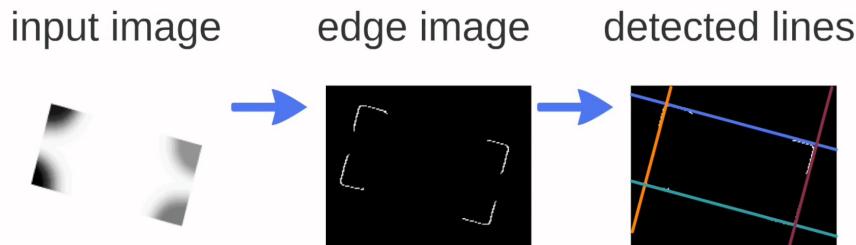
La transformée (ou transformation) de Hough est une technique de reconnaissance de formes inventée en 1962 par Paul Hough.

L'application la plus simple permet de détecter les lignes/droites présentes dans une image mais des modifications peuvent être apportées à cette technique pour détecter d'autres formes géométriques (cercles, ellipses...) : c'est la transformée généralisée de Hough développée par Richard Duda et Peter Hart en 1972.

Cette technique est devenue un outil standard dans le domaine de la vision artificielle.

2.2.2 Principe

Le problème posé est celui de la recherche et de la détection de lignes qui seraient éventuellement présentes dans une image à analyser. Le principe de la transformée de Hough est qu'il existe un nombre infini de lignes qui passent par un point, dont la seule différence est la pente a , l'orientation, l'angle... Le but de la transformée est de déterminer lesquelles de ces lignes sont les plus présentes dans l'image analysée.

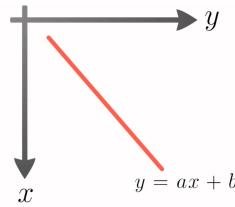


2.2.3 Première approche

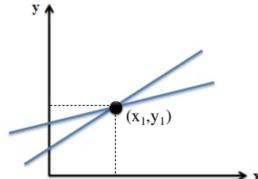
2.2.3.1 Représentation d'une droite

La formule la plus simple représentant une droite dans le plan est son équation cartésienne et réduite $y = ax + b$ (avec $a, b \in \mathbb{R}$) où

$$\begin{cases} a \text{ est la pente de la droite} \\ b \text{ est l'ordonnée à l'origine} \end{cases}$$

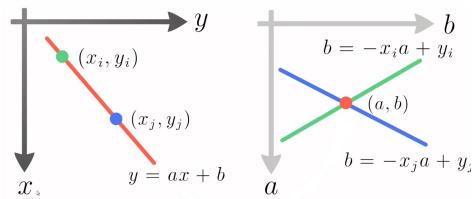


Toutes les droites passant par un point de coordonnées (x_1, y_1) fixées ont la forme $y_1 = ax_1 + b$ pour différentes valeurs de a et b .



2.2.3.2 Principe détaillé

On va changer le repère en remplaçant x par a et y par b . Chaque droite dans l'espace (x, y) sera transformée en un point dans l'espace (a, b) , espace de Hough. Inversément, chaque point dans l'espace (x, y) sera transformé en une droite d'équation $b = -ax + y$ dans l'espace de Hough.



Pour un point A , toutes les droites passant par ce point correspondent à une seule droite a dans l'espace de Hough.

Pour un point B , toutes les droites passant par ce point correspondent à une seule droite b dans l'espace de Hough.

Ces 2 faisceaux de droites dans l'espace (x, y) ont en commun l'unique droite qui relie A et B .

Cette droite correspond, dans l'espace de Hough, au point qui est l'intersub-section des 2 droites a et b .

Cette intersection, ce point contient les paramètres de la droite recherchée.

Tous les points situés sur une même droite c dans l'espace (x, y) sont représentés par des droites passant toutes par un même point C dans l'espace de Hough.

Les coordonnées de ce point $C(x_c, y_c)$ contiennent les paramètres recherchés de la droite $c \equiv y = x_c x + y_c$

Il existe un principe de vote : on utilise un accumulateur où chaque point peut voter pour une droite particulière et les droites recevant le plus de votes sont conservées.

On ne va cependant pas développer ce principe ici.

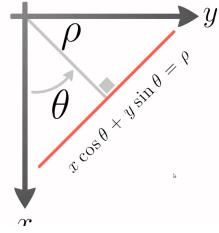
Cette représentation $y = ax + b$ pose un problème pour les droites verticales. Voilà pourquoi la représentation suivante, qui règle ce problème, est la plus utilisée.

2.2.4 Approche trigonométrique

2.2.4.1 Représentation polaire d'une droite

Une droite peut aussi être représentée par la formule $y \sin \theta + x \cos \theta = \rho$ (avec $\theta \in [-\pi/2; \pi/2]$ et $\rho \in [0; d]$ où d est la longueur de la diagonale de l'image) où

$$\begin{cases} \theta \text{ est l'angle entre l'axe } x \text{ et le vecteur } \vec{\rho} \\ \rho \text{ est la distance la plus courte entre l'origine du repère et la droite} \end{cases}$$



2.2.4.2 Principe détaillé

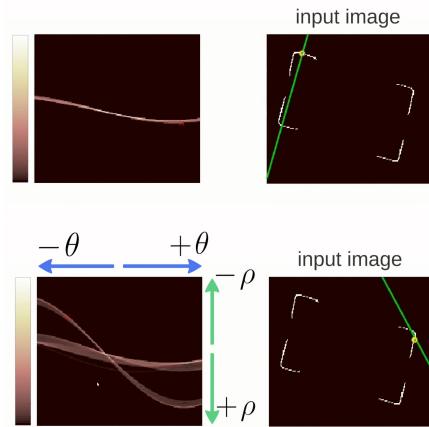
On change également le repère en remplaçant x par θ et y par ρ :
Chaque droite dans l'espace (x, y) sera transformée en un point dans l'espace (ρ, θ) .

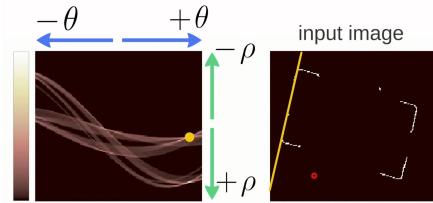
Chaque point dans l'espace (x, y) sera transformé en une courbe, une sinusoïde dans l'espace (ρ, θ) .

Les points d'intersubsection des sinusoïdes dans l'espace (ρ, θ) sont utilisés pour trouver les droites dans l'espace (x, y) .

La transformée de Hough fait ensuite appel à un algorithme qui regarde tous les points et imagine toutes les valeurs possibles pour ρ et θ .

On observe, dans l'espace (ρ, θ) , des pics pour les valeurs qui reviennent le plus souvent et l'algorithme va définir les droites dans l'espace (x, y) et donc les lignes détectées dans l'image.





2.2.5 Conclusion

La transformée de Hough est un outil efficace pour trouver les lignes dans une image.

Il existe d'autres transformées de Hough pour extraire d'autres formes. Elle est utilisée dans plusieurs applications : détection de routes dans les images prises par satellite, lecture de codes barres...

2.3 Détection d'ensembles d'une même couleur

Une autre approche pour détecter des pièces de monnaie sur une image est d'exploiter le contraste entre les pièces elles-mêmes et le matériau sur lequel elles sont placées. Dans l'image suivante, par exemple, les pièces sont jaune ou rouge foncé alors que le fond est blanc.

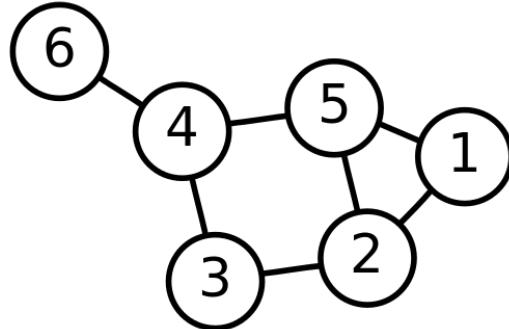


Nous allons développer un algorithme permettant d'utiliser la couleur du fond pour isoler les pièces. Et nous allons l'appliquer à cette image afin d'en extraire les pièces.

3 Un algorithme en profondeur

3.1 Généralités sur les graphes

Un graphe est un ensemble de points appelés *sommets* (*vertices* en anglais) et de connections appelées *arêtes* (*edges* en anglais). Voici par exemple un graphe de 6 sommets et de 7 arêtes :



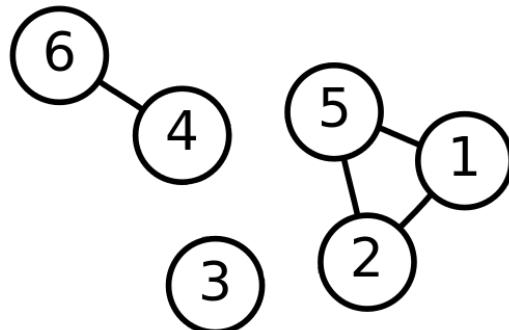
Dans ce cas-ci, les points sont 1, 2, 3, 4, 5, 6 et les connections sont $(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6)$.

D'un point de vue mathématique, le graphe G peut se définir ainsi :

$$\begin{aligned} G &= (V, E), \\ V &= \{1, 2, 3, 4, 5, 6\}, \\ E &= \{(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6)\}. \end{aligned}$$

3.2 Composantes connexes

Le graphe ci-dessus est un exemple de graphe *connexe*, c'est-à-dire un graph dont tout point est relié à tout point. Il existe aussi des graphes non connexes. Les différentes parties connexes du graphe sont alors appelées des *composantes connexes*. Le graphe ci-dessous est ainsi composé de 3 composantes connexes. Il est clair que le graphe est non connexe car le point 1 n'est par exemple pas relié au point 3.



3.3 Parcours d'un graphe

3.3.0.1

Voyons maintenant comment on peut détecter les différentes composantes connexes d'un graphe. Pour ce faire, commençons par remarquer la transitivité de la définition d'une composante connexe. En effet, s'il existe un chemin de u à v et de u à w , il existe forcément un chemin allant de v à w et inversement. Il suffit en effet de partir de v , de se rendre à u et de rejoindre w depuis u . Cette propriété peut sembler anodine, mais elle affirme que pour détecter une composante connexe, il suffit de partir d'un sommet quelconque et de parcourir le graphe en cataloguant tous les sommets atteignables.

3.3.0.2

Un algorithme classique pour parcourir les graphes est le *depth-first search* (DFS) ou encore *parcours en profondeur*, appelé ainsi parce qu'il va aussi loin qu'il peut avant de revenir en arrière.

On part d'un sommet u . Si le sommet u n'a pas encore été visité, on le marque comme visité. Ensuite, pour chaque sommet v adjacent à u , on reprend l'algorithme récursivement. En C++, cela donne :

```
1 const int N; // nombre de sommets (numérotés de 0 à N - 1)
2
3 // Pour chaque sommet, on garde une liste des sommets adjacents
4 vector<int> adjacent_vertices[N];
5
6 // Au départ, aucun des sommets n'a été visité
7 bool visited[N] = {false};
8
9 void dfs(int u) {
10    if (!visited[u]) { // Si u n'a pas été visité
11        visited[u] = true;
12        // Pour chaque sommet v adjacent
13        for (int v : adjacent_vertices[u]) {
14            dfs(v); // On continue récursivement
15        }
16
17        // Cataloguer u d'une façon ou d'une autre
18        /* ... */
19    }
20}
```

Listing 1 – DFS

3.3.0.3

Cet algorithme ne parcourt qu'une seule composante connexe. Afin de détecter toutes les composantes connexes, on peut essayer de lancer l'algorithme depuis chaque sommet. Si l'algorithme n'a pas encore été lancé depuis un sommet u , c'est que u appartient à une nouvelle composante connexe.

```
1 // Pour chaque sommet u
2 for( int u = 0; u < N; ++u) {
3     if( !visited [u]) {
4         // Nouvelle composante connexe
5         // donc on appelle l'algorithme.
6         dfs (u);
7     }
8 }
```

Listing 2 – Composantes connexes

3.3.0.4

Ce code parcourt bien tous les sommets mais il ne permet toujours pas de déterminer la composante connexe d'un sommet donné. Résolvons ce problème : plutôt que d'enregistrer si un sommet a été visité ou non, on peut directement enregistrer à quelle composante connexe il appartient.

```
1 // Au départ, aucun des sommets n'a été visité (cc[u] = -1  ∀u)
2 int cc [N];
3
4 void dfs( int u,  int id) {
5     if( cc [u] == -1) { // Si u n'a pas été assigné
6         cc [u] = id ;
7         // Pour chaque sommet v adjacent
8         for( int v : adjacent _vertices [u]) {
9             dfs(v,  id); // On continue récursivement
10        }
11    }
12 }
13
14 // On initialise à -1
15 for( int u = 0; u < N; ++u) cc [u] = -1;
16
17 // Composante connexe actuelle
18 int id = 0;
19 // Pour chaque sommet u
20 for( int u = 0; u < N; ++u) {
```

```
21 |     if (cc[u] == -1) {  
22 |         // Nouvelle composante connexe  
23 |         // donc on appelle l'algorithme.  
24 |         dfs(u, id++);  
25 |     }  
26 }
```

Listing 3 – Composantes connexes