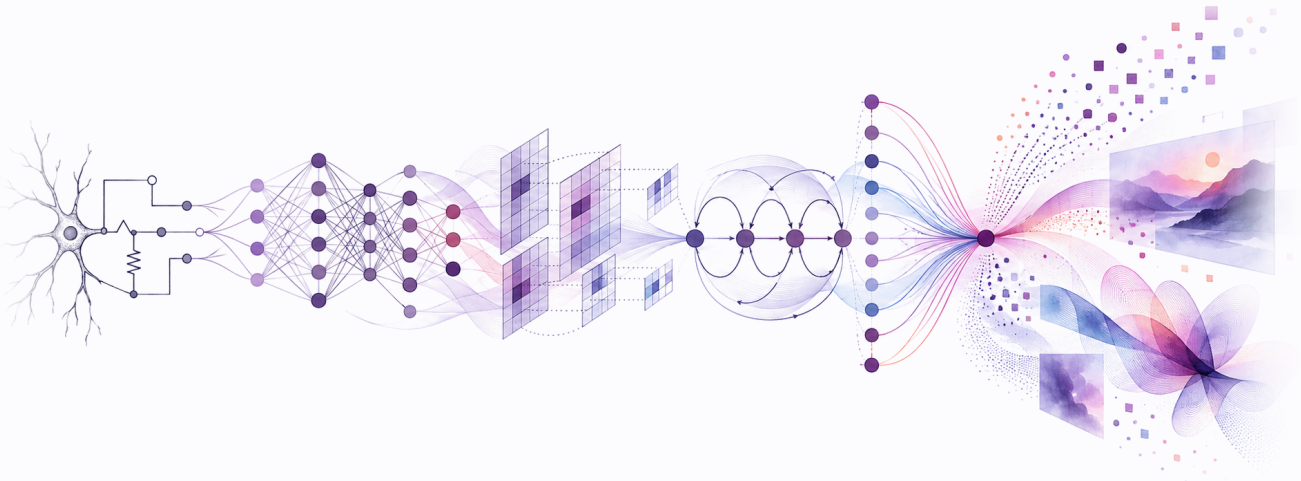
L’histoire de l’IA

Des premiers neurones aux IA génératives



Du neurone formel de 1943 aux modèles génératifs d’aujourd’hui.

Frédéric Ros

Cofinancé par l’Union européenne

L'ESSENTIEL À RETENIR

L'intelligence artificielle a connu des **naissances, des morts et des renaissances**. Ce n'est pas une suite de miracles indépendants : c'est une chaîne de causes et de conséquences. Chaque technologie naît pour résoudre un problème concret, diffuser une expertise rare, piloter une machine en douceur, donner une mémoire à un réseau — puis se heurte à ses propres limites, qui deviennent à leur tour la raison d'être de la technologie suivante. Voici cette histoire, de **1943 à 2026**.

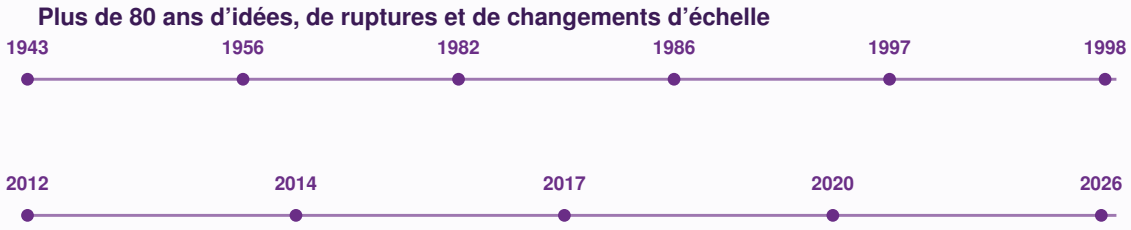


Figure 1. Quelques jalons pour se repérer. Cette chronologie simplifiée sert de carte de lecture ; plusieurs familles d'IA coexistent encore aujourd'hui.

1943–1958 : l'IA naît sur le papier

L'intelligence artificielle commence comme une idée philosophique : et si une machine pouvait imiter, même de très loin, l'intelligence humaine ? En pleine Seconde Guerre mondiale, un neurophysiologiste, Warren McCulloch, et un logicien, Walter Pitts,[1] posent les bases théoriques des réseaux de neurones. Ils n'ont pas d'ordinateurs modernes à disposition, mais ils ont des équations : ils imaginent un neurone artificiel qui fonctionnerait, en simplifié, comme un neurone biologique. Ce dernier reçoit des signaux électriques par ses dendrites, les combine, et s'il dépasse un certain seuil d'excitation, il s'active et transmet à son tour un signal. Le neurone artificiel de McCulloch et Pitts fait le même calcul, mais avec des nombres : il reçoit plusieurs valeurs, les multiplie chacune par un **poids**, l'importance qu'on leur accorde, additionne le tout, et s'active si la somme dépasse un seuil. C'est, au fond, ce que vous faites lorsque vous décidez de sortir un soir en pesant plusieurs critères (la météo, votre fatigue, la présence de vos amis) : chacun compte plus ou moins, et si le total franchit un certain seuil de motivation, vous sortez.

Il faut cependant se méfier de l'analogie avec le cerveau. Le terme « réseau de neurones » est resté dans le vocabulaire, mais un neurone artificiel est une simplification radicale d'un neurone biologique, lui-même bien plus complexe (chimie, plasticité, timing des signaux). Le réseau de neurones artificiel emprunte une métaphore, pas une reproduction fidèle du cerveau.

À l'époque, ces idées restent purement théoriques : les ordinateurs sont trop lents et trop contraints pour les exploiter à grande échelle. C'est un peu comme disposer du plan détaillé d'un moteur sans avoir encore les matériaux pour le construire. En 1950, Alan Turing publie *Computing Machinery and Intelligence*[2] et propose ce qui deviendra le **test de Turing** : plutôt que de définir abstraitement l'intelligence, il propose un critère pratique, si un humain ne peut pas distinguer une machine d'un humain lors d'une conversation écrite, la machine

peut être considérée comme se comportant intelligemment. Puis en 1956, lors de l'atelier de Dartmouth, l'expression « **intelligence artificielle** » s'impose comme nom d'un nouveau domaine de recherche.

Alan Turing, 1951. En proposant de juger une machine par son comportement observable plutôt que par une définition abstraite de la pensée, Turing déplace profondément la question de l'intelligence artificielle.
Photo Elliott & Fry, via Wikimedia Commons, domaine public.

Dartmouth College. C'est lors de l'atelier de l'été 1956 que l'expression « intelligence artificielle » devient le nom d'un champ de recherche. Le bâtiment est photographié ici plusieurs décennies plus tard.
Photo Daderot, Wikimedia Commons, CC0 1.0.

Figure 2. Deux repères fondateurs : une question, puis un domaine scientifique.

« Une machine peut-elle penser ? »
Alan Turing, 1950

En 1958, cette intuition devient un système capable d'apprendre avec le **perceptron** de Rosenblatt[6]. La nouveauté décisive est la séparation entre deux moments : pendant l'entraînement, les erreurs servent à corriger les poids ; une fois ces poids fixés, le même calcul classe des observations que le système n'a jamais vues.

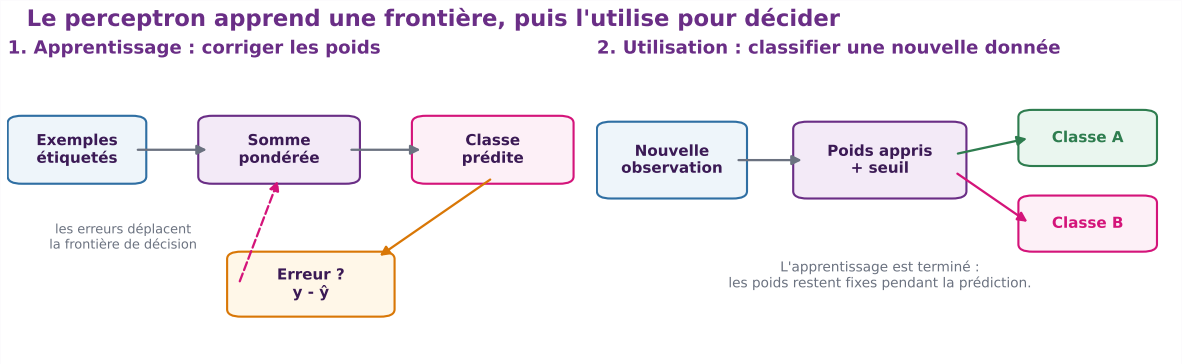


Figure 3. Apprendre, puis prédire. À gauche, les erreurs commises sur des exemples étiquetés corrigent les poids du perceptron. À droite, l'apprentissage est terminé : les poids restent fixes et permettent de classer une nouvelle observation. Schéma original d'après Rosenblatt (1958)[6].

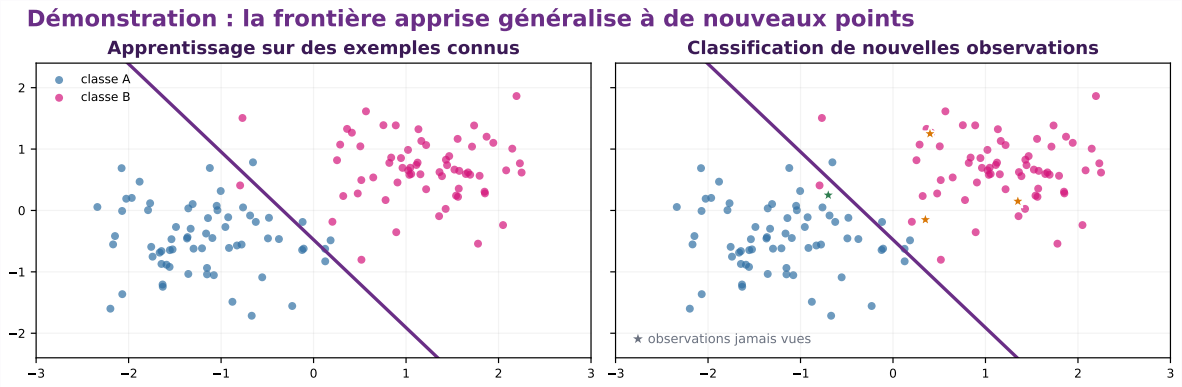


Figure 4. Démonstration reproductible de classification. Le perceptron apprend une frontière linéaire à partir de deux groupes d'exemples connus, puis l'utilise pour attribuer une classe aux étoiles, qui représentent de nouvelles observations. Simulation originale (graine 4) d'après Rosenblatt (1958)[6] ; code : `scripts/generate_demo_figures.py`.

1960–1990 : les hivers de l'IA

L'IA a souvent trop promis, trop tôt. Les premiers programmes, comme ELIZA, impressionnent par leur nouveauté mais restent très limités. Lorsque les résultats ne suivent pas les attentes, les financements se tarissent : ce sont les **hivers de l'IA**, des périodes où l'enthousiasme retombe brutalement, comme une mode qui s'essouffle quand elle ne tient pas ses promesses. C'est dans ce contexte en dents de scie que naissent, coup sur coup, trois idées qui vont marquer durablement le domaine : les systèmes experts, la logique floue, et leur mariage, le neuro-flou.

Capturer l'expertise rare : les systèmes experts

Dans les années 1970, un constat s'impose aux chercheurs comme aux industriels : les meilleurs experts d'un domaine, un médecin chevronné, un ingénieur qui connaît sa chaîne de production par cœur, un géologue capable de repérer un gisement d'un coup d'œil, possèdent un savoir précieux, mais rare, coûteux à former, et difficile à transmettre. Un grand spécialiste ne peut être physiquement présent partout. La question se pose alors avec une simplicité presque brutale : peut-on capturer le raisonnement d'un expert dans un programme, pour le rendre disponible à plus grande échelle ? C'est de cette ambition que naissent, dans les années 1980, les **systèmes experts**.

Leur fonctionnement reprend, presque à l'identique, ce que fait n'importe qui en suivant une recette de cuisine ou un arbre de dépannage, « si l'appareil ne s'allume pas, vérifiez d'abord qu'il est branché ; si oui, vérifiez le fusible ». On applique, étape par étape, des règles explicites écrites par quelqu'un d'autre. Concrètement, un système expert repose sur trois éléments : une **base de connaissances**, qui décrit les faits et les règles du domaine (« SI condition A ET condition B ALORS conclusion C ») ; un **moteur d'inférence**, qui applique ces règles aux faits disponibles pour produire de nouvelles conclusions ; et une **interface**, qui permet d'interroger le système. Ces règles ne sortent pas de nulle part : un **cogniticien**, un spécialiste chargé d'extraire et de formaliser le savoir de l'expert, passe des mois à interviewer le spécialiste humain pour transformer son intuition en un ensemble de règles écrites noir sur blanc.

Domaine	Système	Ce qu'il faisait
Médecine	MYCIN[4] (Stanford, 1970s)	Proposait un diagnostic d'infections bactériennes et suggérait un traitement antibiotique, en interrogeant l'utilisateur comme le ferait un interne consultant un médecin senior.
Industrie	XCON[5] (Digital Equipment, 1980)	Configurait automatiquement des commandes complexes d'ordinateurs en vérifiant la compatibilité des composants, une tâche réservée jusque-là à quelques techniciens expérimentés.
Géologie	PROSPECTOR	Évaluait le potentiel minier d'un site à partir d'observations de terrain, en s'appuyant sur les règles qu'auraient utilisées des géologues chevronnés.

Table 1. Trois systèmes experts pionniers et leurs domaines d'application

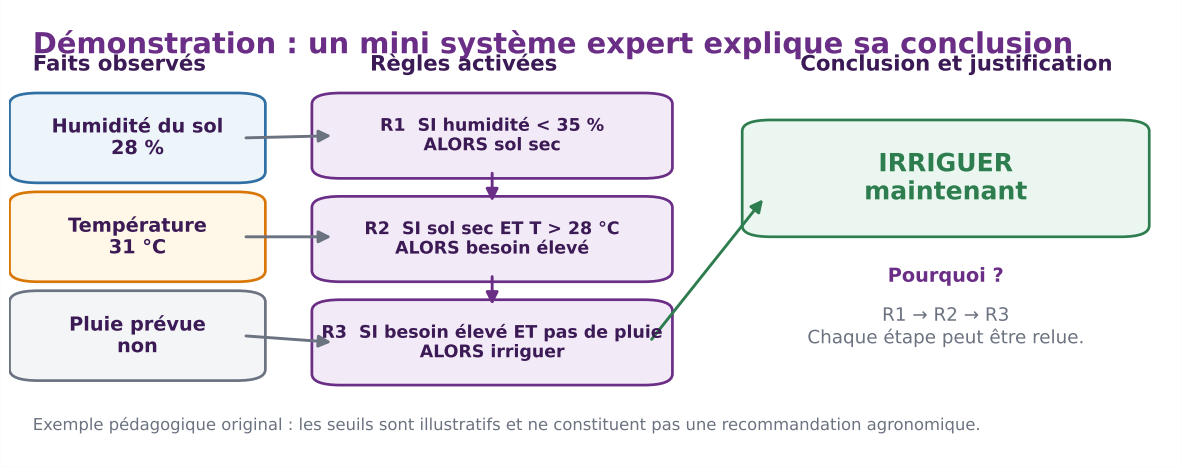


Figure 5. Démonstration d'un mini système expert. Trois faits activent successivement trois règles explicites jusqu'à une décision d'irrigation. La chaîne R1 → R2 → R3 fournit immédiatement une justification lisible, comme dans les mécanismes d'explication de MYCIN[4]. Exemple original ; les seuils sont volontairement illustratifs et ne constituent pas une recommandation agronomique.

Pour la première fois, une expertise rare pouvait être diffusée au-delà de la seule personne qui la détenait. Et surtout, le raisonnement du système restait **traçable** : on pouvait lui demander « pourquoi as-tu conclu cela ? » et remonter la chaîne de règles utilisées, une transparence que les réseaux de neurones modernes peinent encore à offrir.

Piloter en douceur : la logique floue

À la même époque, un autre problème occupe les ingénieurs du contrôle-commande. Un système de pilotage classique raisonne en tout ou rien : « si température $>$ 25°C, ALORS éteindre le chauffage ». Résultat, des à-coups permanents, le chauffage s'allumant et s'éteignant sans cesse autour du seuil. Or, dans la vraie vie, personne ne raisonne ainsi : « il fait chaud » n'est pas une question de tout ou rien, 28°C, c'est plutôt chaud, 35°C, c'est très chaud, et 20°C, c'est un peu tiède. En 1965, Lotfi Zadeh[3] propose de laisser une machine raisonner avec ces degrés d'appartenance plutôt qu'avec des seuils binaires : c'est la naissance de la **logique floue** (*fuzzy logic*).

Le principe tient en trois étapes. D'abord, la **fuzzification** : une valeur numérique est traduite en degrés d'appartenance à différentes catégories (« un peu chaud », « très chaud »). Ensuite, des **règles floues** combinent ces catégories. Enfin, la **défuzzification** reconvertit le résultat flou en une commande concrète. C'est exactement ce que fait un thermostat qui, en s'approchant de la température cible, réduit progressivement le chauffage plutôt que de basculer brutalement entre marche et arrêt, ou ce que vous faites vous-même en conduisant : vous ne freinez jamais à fond ou pas du tout, vous dosez la pression selon la distance qu'il reste.

Domaine	Application concrète
Électroménager	Machines à laver et climatiseurs japonais (Panasonic, Hitachi, années 1990) ajustant en continu eau, lessive ou froid selon le linge ou la pièce.
Transport	Régulation de la conduite du métro de Sendai (Japon, 1987) : accélérations et freinages doux, plus économes qu'un pilotage tout-ou-rien.
Industrie	Pilotage de fours industriels et de procédés chimiques, où température et pression évoluent continûment.
Agriculture	Régulation de serres (température, hygrométrie, irrigation), où les besoins de la plante varient graduellement.

Table 2. Applications historiques de la logique floue

Le bénéfice est net : des systèmes de contrôle plus stables, plus économes en énergie, et surtout plus faciles à concevoir à partir du simple bon sens d'un opérateur expérimenté, sans avoir besoin d'un modèle mathématique complet du système à piloter.

Marier la règle et l'apprentissage : le neuro-flou

Vers la fin des années 1980, un problème symétrique apparaît des deux côtés à la fois. Les systèmes experts et la logique floue fonctionnent bien, à condition qu'un humain sache déjà formuler les règles. Mais que faire quand personne ne sait exprimer précisément son savoir-faire, quand l'expertise est davantage une intuition acquise par l'expérience qu'un ensemble de règles conscientes ? À l'inverse, les réseaux de neurones de l'époque savent apprendre à partir de données, mais restent des boîtes noires, incapables d'expliquer pourquoi ils arrivent à telle conclusion. L'idée du **neuro-flou** (*neuro-fuzzy*) est de faire coopérer les deux mondes : utiliser un réseau de neurones pour apprendre automatiquement les règles floues à partir de données, plutôt que de les faire écrire à la main par un expert. La structure reste celle de la logique floue, catégories, règles, défuzzification — ce qui garde le raisonnement lisible ; mais les paramètres de ces règles (à partir de quelle température considère-t-on qu'« il fait chaud » ?) sont ajustés automatiquement par apprentissage, à partir d'exemples observés. C'est un

peu comme un apprenti cuisinier qui part de la structure d'une recette, les grandes étapes, les catégories d'ingrédients, mais qui, en goûtant des centaines de plats, affine seul les proportions exactes, sans qu'un chef ait à les lui dicter au gramme près.

Cette approche trouve un écho durable en maintenance industrielle, où elle permet de diagnostiquer des pannes sur des équipements complexes en affinant les règles de détection d'anomalies à partir de l'historique des capteurs, tout en restant capable d'expliquer sur quels signaux le système s'est appuyé ; en médecine, pour ajuster les seuils de règles cliniques connues aux variations individuelles des patients ; et dans le pilotage de procédés industriels, pour adapter automatiquement les règles de contrôle à l'usure des machines ou à la variabilité des matières premières. Le compromis qu'offre le neuro-flou, performance grâce à l'apprentissage, explicabilité grâce à la structure floue, reste aujourd'hui recherché face aux réseaux profonds, souvent très performants mais peu capables de justifier leurs décisions.

> **POINT DE VIGILANCE**
>
> **Pourquoi un deuxième hiver ?** À la fin des années 1980, les attentes placées dans les systèmes experts deviennent difficiles à satisfaire : leur développement et leur maintenance sont coûteux, une base de règles trop grande devient illisible et incohérente (un peu comme un règlement intérieur qui, à force d'exceptions, ne veut plus rien dire), et le système ne découvre jamais seul de nouvelles règles à partir des données. Les investissements chutent fortement. La période allant approximativement de la fin des années 1980 au début des années 1990 est le **deuxième hiver de l'IA**. Mais ces technologies n'ont pas disparu pour autant : elles continuent aujourd'hui, discrètement, de faire tourner des machines à laver, des ascenseurs et des lignes de production, une IA silencieuse, moins spectaculaire que les grands modèles de langage, mais toujours largement utilisée.

1982 : deux autres façons de faire apprendre un réseau

L'histoire des réseaux de neurones ne conduit pas directement du perceptron au deep learning. Au début des années 1980, deux familles très différentes montrent déjà qu'un réseau peut faire autre chose que simplement classer une entrée : il peut **mémoriser** ou **s'organiser tout seul**.

Hopfield : retrouver un souvenir à partir d'un fragment

En 1982, John Hopfield[17] propose un réseau récurrent dans lequel les neurones sont reliés les uns aux autres par des connexions symétriques. L'idée importante n'est pas seulement l'architecture, mais le comportement collectif du réseau. On lui présente plusieurs motifs à mémoriser ; ensuite, si on lui fournit une version incomplète ou bruitée de l'un de ces motifs, son état évolue progressivement vers une configuration stable correspondant au souvenir appris. C'est une **mémoire associative** : quelques éléments peuvent suffire à rappeler l'ensemble, un peu comme quelques notes d'une mélodie permettent parfois de reconnaître immédiatement la chanson.

Cette idée introduit aussi une notion particulièrement féconde, celle de **paysage d'énergie**. Les souvenirs correspondent à des vallées ; à partir d'un état imparfait, le réseau descend vers

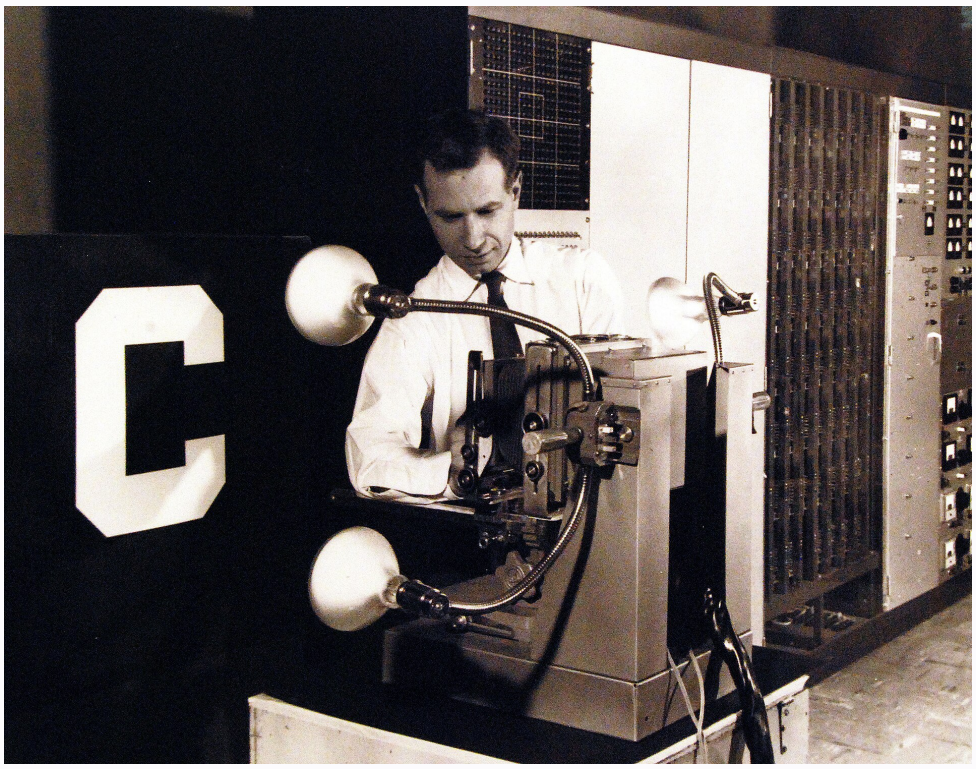


Figure 6. Le Mark I Perceptron. À la fin des années 1950, le perceptron de Frank Rosenblatt fait passer le neurone artificiel du papier à la machine. Son dispositif expérimental apprend à reconnaître des motifs à partir d'une « rétine » de photocellules. *Photo : National Museum of the U.S. Navy, domaine public (PD US Navy), via Wikimedia Commons.*

une vallée stable. Cette vision des réseaux comme systèmes dynamiques aura une influence durable, bien au-delà du modèle de 1982.

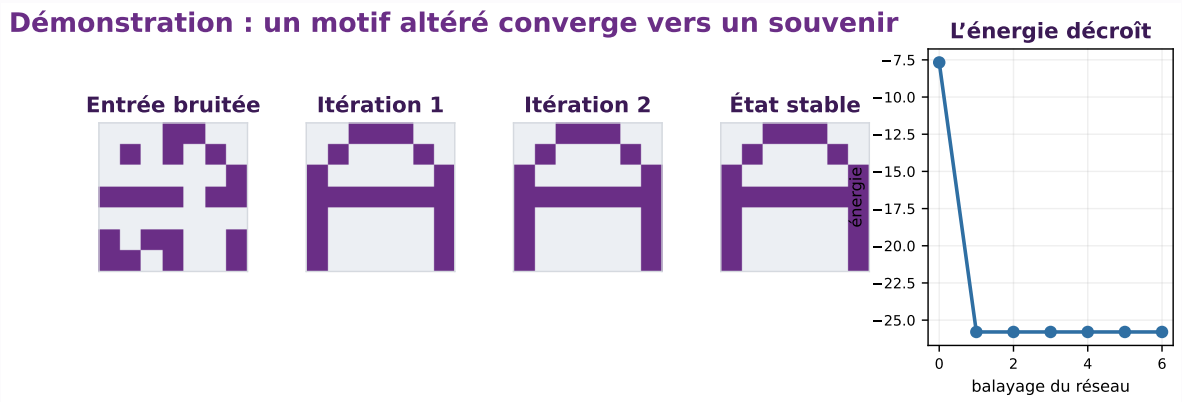


Figure 8. Démonstration reproductible d'une mémoire associative. Dix pixels d'un motif 7×7 mémorisé ont été inversés. Avec une règle de Hebb et des mises à jour asynchrones, le réseau retrouve le motif stable tandis que son énergie décroît. Simulation originale (graine 7) d'après Hopfield (1982)[16] ; code : `scripts/generate_demo_figures.py`.

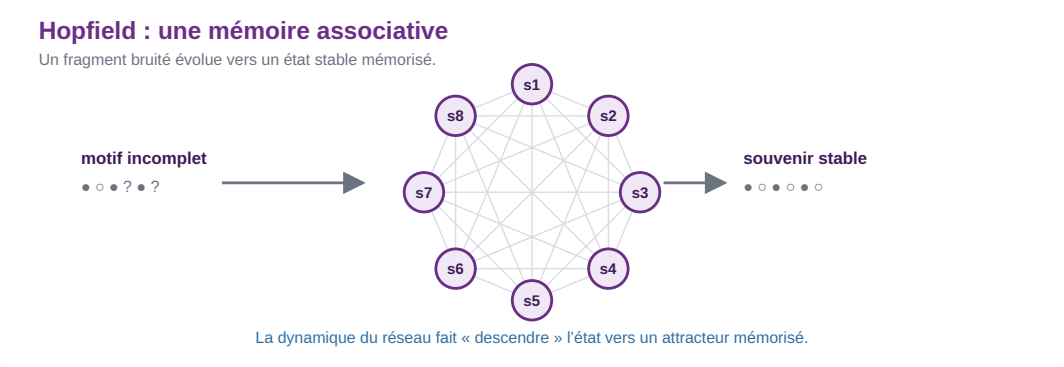


Figure 7. Réseau de Hopfield : les connexions récurrentes permettent de faire évoluer un motif incomplet vers un état mémorisé. Schéma original d'après Hopfield (1982).

Kohonen : transformer des données complexes en carte

La même année, Teuvo Kohonen[18] décrit un principe très différent : la **carte auto-organisatrice**, ou SOM (*Self-Organizing Map*). Ici, aucune étiquette n'indique au réseau la bonne réponse. Les neurones, disposés sur une grille, entrent en compétition pour représenter chaque observation. Le neurone qui ressemble le plus à l'entrée gagne ; lui et ses voisins se déplacent légèrement vers cette observation. Répété sur de nombreuses données, ce mécanisme organise progressivement la carte de sorte que des observations similaires occupent des régions voisines.

Figure 9. Teuvo Kohonen dans les années 1980. Ses cartes auto-organisatrices donnent une traduction visuelle à une idée devenue centrale en apprentissage non supervisé : découvrir la structure des données sans disposer d'étiquettes. *Photo : Teknillinen korkeakoulu / Aalto University, via Wikimedia Commons, CC BY 4.0.*

Le résultat est particulièrement intuitif : des données comportant des dizaines de variables peuvent être projetées sur une carte en deux dimensions qui en préserve autant que possible les voisinages. Les SOM ont ainsi joué un rôle important dans le **clustering**, l'exploration et la visualisation de données. Elles rappellent surtout qu'apprendre ne signifie pas toujours prédire une étiquette : une machine peut aussi apprendre la **structure** d'un ensemble de données sans

professeur.

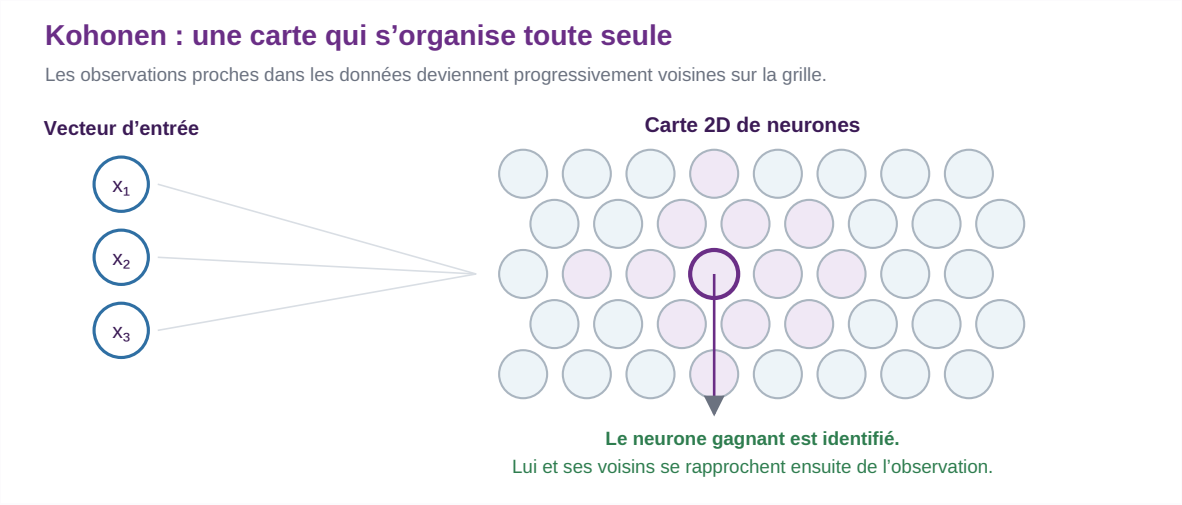


Figure 10. Carte de Kohonen : chaque observation active un neurone gagnant et entraîne son voisinage ; la grille s'organise progressivement en une carte topologique des données. Schéma original d'après Kohonen (1982).

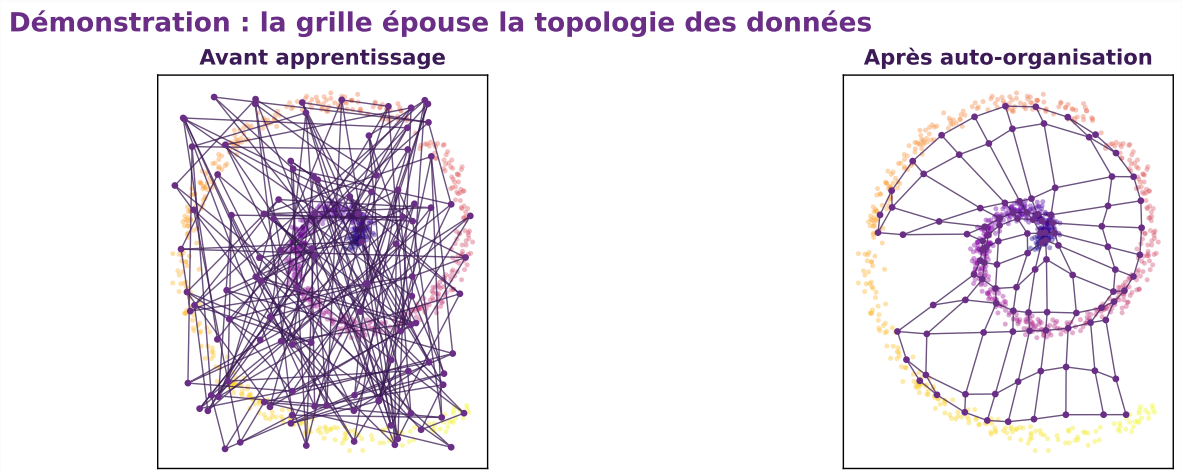


Figure 11. Démonstration reproductible d'une carte auto-organisatrice. Une grille de 9×12 prototypes, initialement désordonnée, apprend la géométrie d'un nuage en spirale sans utiliser les couleurs comme étiquettes. Simulation originale (7,000 tirages, graine 12) d'après Kohonen (1982)[17] ; code : `scripts/generate_demo_figures.py`.

Donner une mémoire, puis un apprentissage, aux réseaux

Un neurone artificiel isolé, le **perceptron**, proposé dès 1958 par Frank Rosenblatt[6], ne peut résoudre que des problèmes très simples. La véritable puissance viendra de l'empilement de plusieurs neurones en couches, et surtout d'une méthode pour les entraîner automatiquement : la **rétropropagation du gradient**, popularisée au milieu des années 1980[7]. Le principe est le suivant : le réseau fait une prédiction (reconnaître un chiffre manuscrit, par exemple), on la compare à la bonne réponse pour obtenir une erreur, puis cette erreur est propagée en arrière à travers le réseau, couche par couche, pour déterminer la contribution de chaque

poids à l'erreur commise et l'ajuster légèrement. C'est comme apprendre à viser à la fléchette les yeux bandés, avec un ami qui vous dit seulement « trop à gauche » ou « trop bas » après chaque lancer : vous ne savez pas exactement pourquoi vous avez raté, mais vous ajustez votre geste en fonction de ce retour, encore et encore, jusqu'à ce qu'il s'affine. Un réseau de neurones apprend de la même façon, en essais, erreurs et petits ajustements répétés, mais des millions de fois, très rapidement, et de façon parfaitement mathématique. C'est ce mécanisme qui permettra au *deep learning* de véritablement décoller à partir des années 2000–2010.

1990–2010 : le retour progressif de l'apprentissage

1998 : LENET-5 ET LES PREMIERS CNN MODERNES

Yann LeCun et ses collaborateurs présentent **LeNet-5**[8], une architecture emblématique de réseau de neurones convolutif, utilisée pour la reconnaissance de chiffres manuscrits.

Avant LeNet-5, pour qu'un programme reconnaisse un chiffre ou un visage dans une image, il fallait qu'un ingénieur définisse à la main les caractéristiques à repérer — « un 8 a deux boucles », « un visage a deux yeux espacés de tant de pixels ». Ce travail est long, fragile (il suffit d'un angle de prise de vue différent pour que tout s'effondre) et ne se généralise pas d'une tâche à l'autre. LeNet-5 répond à une question radicalement différente : peut-on laisser le réseau découvrir lui-même les caractéristiques utiles, directement à partir d'exemples d'images ? Le réseau convolutif procède comme un détective qui examine une scène par petits morceaux : il ne regarde pas la photo entière d'un seul coup, mais repère d'abord des indices locaux, une trace, une courbe, un contour, grâce à des filtres qui parcourent l'image, puis combine progressivement ces indices en une conclusion globale. Une étape de sous-échantillonnage réduit peu à peu la représentation tout en conservant l'essentiel, un peu comme un résumé qui garde le fil sans tous les détails ; enfin, une couche de classification assemble les caractéristiques apprises pour trancher. Avec sept couches entraînables et environ 60 000 paramètres, à comparer aux centaines de milliards des grands modèles actuels, LeNet-5 obtient des résultats remarquables, pour l'époque, sur la reconnaissance de chiffres manuscrits.

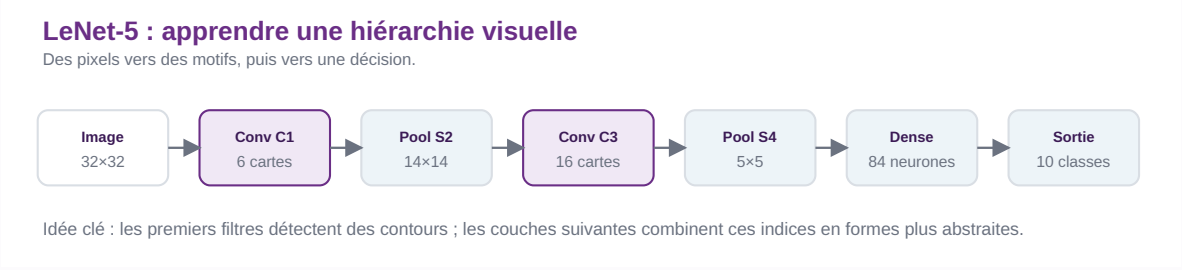


Figure 12. LeNet-5 : la hiérarchie caractéristique d'un CNN, des pixels aux motifs puis à la classe. Schéma original, inspiré de l'architecture publiée par LeCun et al. (1998).

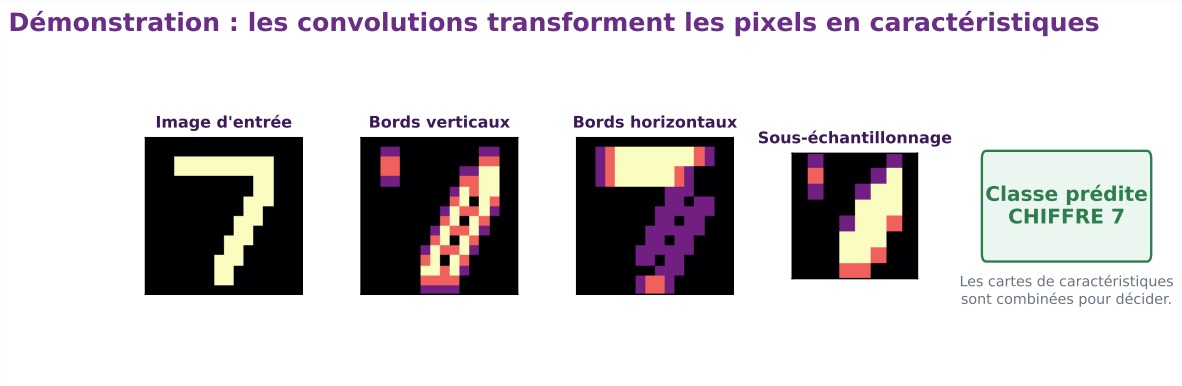


Figure 13. Démonstration visuelle d'une convolution. Des filtres simples font apparaître séparément les bords verticaux et horizontaux d'un chiffre ; le sous-échantillonnage compacte ensuite ces cartes avant la classification. Illustration originale reproductible du principe des CNN d'après LeCun et al. (1998)[8] ; code : `scripts/generate_demo_figures.py`.

SOUS LE CAPOT – lecture avancée facultative

Un CNN en trois gestes. Une **convolution** fait glisser de petits filtres sur l'image et apprend à détecter des motifs locaux, d'abord des bords et des textures, puis des formes plus complexes. Le **pooling** ou une réduction équivalente diminue progressivement la taille de la représentation en conservant les informations les plus utiles. Enfin, un **classifieur** combine les motifs détectés pour produire une décision. L'image simple à garder en tête est celle d'un détective qui inspecte d'abord de petits indices avant de reconstruire la scène complète.

POINT DE VIGILANCE

Pourquoi les CNN n'explosent-ils pas immédiatement ? Les jeux de données disponibles restent limités, la puissance de calcul très inférieure à celle d'aujourd'hui, et les GPU ne sont pas encore utilisés massivement pour l'entraînement. C'est un peu comme avoir une excellente recette, mais ni assez d'ingrédients (les données) ni un four assez puissant (le calcul) pour la préparer à grande échelle. Il faudra attendre **2012 et AlexNet** pour que l'apprentissage profond en vision change brutalement d'échelle.

Le renouveau de l'IA repose alors sur trois piliers qui doivent avancer ensemble pour produire un vrai bond en avant : davantage de **données**, portées par l'essor du Web dans les années 2000 ; davantage de **puissance de calcul** ; et de meilleurs **algorithmes d'apprentissage**. En 1997 déjà, Deep Blue avait battu Garry Kasparov aux échecs[16]. Ce moment spectaculaire mérite cependant d'être bien interprété : Deep Blue n'est pas un ancêtre direct des LLM. Il combine une recherche très rapide dans l'arbre des coups, des fonctions d'évaluation sophistiquées, des bases d'ouvertures et de finales, ainsi qu'un matériel massivement parallèle spécialisé pour les échecs. Lors du match de 1997, il pouvait analyser en moyenne environ **200 millions de positions par seconde**. Sa victoire montre qu'un comportement que nous qualifions d'intelligent peut émerger d'une combinaison de calcul, de recherche et d'expertise codée, sans apprentissage profond moderne.

En 2006, les travaux sur les réseaux profonds relancent l'intérêt pour le *deep learning*. Puis en 2012, AlexNet[10] obtient des résultats spectaculaires au concours ImageNet et marque le véritable tournant.

Kasparov face à Deep Blue, 1997. La revanche en six parties se termine sur le score de 3,5 à 2,5 en faveur de la machine, un événement mondialement médiatisé.
Photographie d'archive IBM fournie avec le projet ; droits de réutilisation à confirmer avant diffusion éditoriale.

Deep Blue. Un exemplaire exposé au Computer History Museum.
Photo Anton Chiang, Wikimedia Commons, CC BY 2.0.

Figure 14. 1997 : la machine bat le champion. Une rupture symbolique majeure, mais fondée sur une IA spécialisée très différente des réseaux génératifs actuels.

L'IDÉE À RETENIR

Pourquoi AlexNet change-t-il vraiment la donne ? Ce n'est pas une idée isolée qui provoque la rupture, mais la rencontre de plusieurs ingrédients devenus suffisamment mûrs au même moment : un CNN plus profond, des fonctions d'activation ReLU efficaces, du *dropout* pour limiter le surapprentissage, un très grand jeu d'images annotées et surtout l'utilisation intensive des **GPU**. AlexNet illustre ainsi un principe récurrent de l'histoire de l'IA : une architecture devient révolutionnaire lorsque les données et le calcul permettent enfin d'en exploiter le potentiel.

Domaine	Application des CNN (à partir des années 2010)
Médecine	Détection de tumeurs sur mammographies et scanners, analyse de grains de beauté suspects, dépistage de rétinopathie diabétique sur des photos de fond d'œil.
Agriculture	Reconnaissance de maladies sur feuilles de culture par simple photo, tri visuel de fruits et légumes, comptage de plants par drone.
Industrie	Contrôle qualité visuel sur chaîne (défauts de soudure, rayures), inspection d'infrastructures (ponts, éoliennes) par drone.

Table 3. Cas d'usage des réseaux convolutifs profonds

Le langage : des règles au récit qui s'oublie

Comprendre et générer du langage pose un défi différent de la reconnaissance d'images : une phrase est une séquence, où l'ordre des mots compte, où le sens d'un mot dépend souvent de mots dits bien avant lui, et où la longueur du texte n'est jamais fixée à l'avance. Cette difficulté a suivi son propre chemin, des règles de grammaire écrites à la main jusqu'au Transformer.

Les toutes premières tentatives de traduction automatique, dès les années 1950–1960, reposent sur des règles linguistiques écrites à la main, des dictionnaires bilingues et des règles de grammaire, un peu comme un système expert appliqué à la langue. Le résultat : des traductions souvent mot à mot, maladroites, incapables de gérer les expressions idiomatiques ou les ambiguïtés (« avocat », le métier ou le fruit ?). Dans les années 1990–2000, la **traduction statistique** change d'approche : plutôt que d'écrire des règles de grammaire, on apprend les correspondances entre langues à partir de grands volumes de textes déjà traduits, comme les débats du Parlement européen. C'est comme apprendre une langue non pas en mémorisant sa grammaire, mais en lisant des milliers de phrases déjà traduites et en devinant, par la pratique, quelles tournures « sonnent juste ». Les premières versions de Google Traduction, lancées en 2006, reposaient sur cette approche.

Restait un problème que ni les règles ni les statistiques ne résolvaient vraiment : donner à la machine une forme de **mémoire**. Un réseau « classique » traite une entrée de taille fixe et n'a aucune notion d'ordre ou de temps. Or une phrase, une série de mesures de capteurs, un signal audio, sont des séquences où ce qui vient avant influence ce qui vient après. À la fin des années 1980 apparaissent les **réseaux de neurones récurrents** (RNN), conçus précisément pour cela : le réseau traite la séquence mot par mot, dans l'ordre, et à chaque étape combine le mot courant avec un état interne, une sorte de résumé de tout ce qu'il a lu jusque-là — qu'il met à jour au fur et à mesure. C'est exactement ce que fait quelqu'un qui lit une phrase mot après mot en se composant un résumé mental, sans pouvoir revenir en arrière relire le début.

Mais ce résumé mental a ses limites, et elles se révèlent vite critiques. Sur un texte long, l'information mentionnée au tout début s'atténue et finit par s'effacer avant que le réseau n'atteigne la fin, comme un message déformé au fil du jeu du téléphone arabe. En 1997, Sepp Hochreiter et Jürgen Schmidhuber proposent les[9] **LSTM** (*Long Short-Term Memory*), une variante de RNN dotée de « portes » qui décident explicitement ce qu'il faut garder, oublier ou laisser passer, un carnet de notes où l'on choisirait consciemment ce qui mérite d'être noté durablement. Les LSTM (et leur variante simplifiée, les GRU) deviennent, dans les années 2000–2010, l'architecture dominante pour traiter texte, parole et séries temporelles : traduction neuronale, reconnaissance vocale, analyse de signaux vitaux en médecine, maintenance prédictive à partir de vibrations ou de températures de capteurs, prévision de rendement agricole à partir de séries météo.

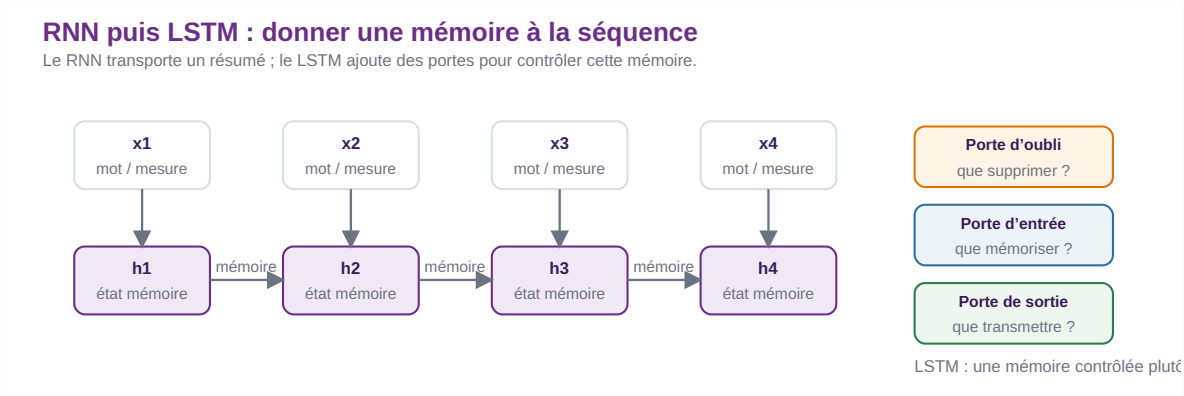


Figure 15. Du RNN au LSTM : le premier transporte un état de mémoire ; le second ajoute des portes qui apprennent quoi conserver, oublier et transmettre.

La même logique s'applique à une série temporelle : pendant l'apprentissage, le réseau découvre

comment le niveau présent, la pluie récente et la saison influencent le niveau suivant. En prévision, son état interne résume les observations passées et est réinjecté pas à pas pour prolonger la courbe au-delà du dernier jour mesuré.

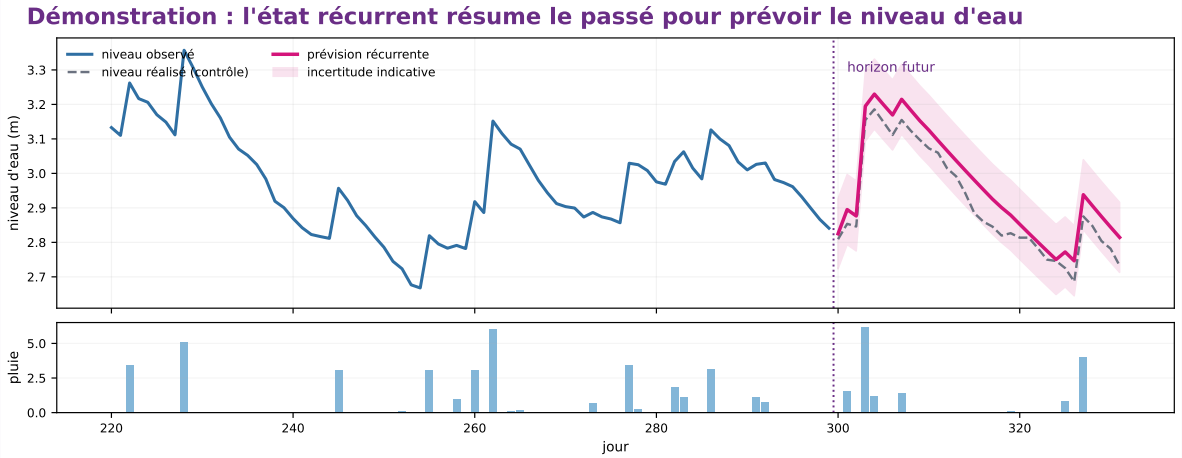


Figure 16. Démonstration reproductible de prévision temporelle. Un prédicteur récurrent minimal apprend sur une série synthétique de niveau d'eau influencée par la pluie et la saison, puis prévoit 32 jours par récurrence. La courbe grise, masquée au modèle, permet de contrôler la prévision ; les pluies futures sont supposées connues, comme dans un scénario alimenté par des prévisions météorologiques. Illustration originale (graine 19) du principe des RNN/LSTM[9] ; code : `scripts/generate_demo_figures.py`.

POINT DE VIGILANCE

L'impasse des RNN et LSTM. Même avec leurs portes, les LSTM peinent à préserver un lien entre un mot en tout début de texte et un mot très éloigné à la fin. Et surtout, un RNN doit lire le mot 1, puis le mot 2, puis le mot 3… dans l'ordre, sans pouvoir paralléliser ce calcul sur plusieurs mots à la fois. Sur de très longs textes et avec les volumes de données modernes, cela devient extrêmement lent à entraîner. C'est précisément pour dépasser ces deux limites, la lenteur du traitement séquentiel et l'amnésie sur les dépendances longues, que naît en 2017 le **Transformer**.

En 2017, une équipe de Google (Vaswani et al.)[13] pose une question presque provocatrice : a-t-on vraiment besoin de lire une phrase dans l'ordre ? Et si, à la place, chaque mot pouvait regarder directement tous les autres mots de la phrase en une seule fois, pour décider lesquels sont pertinents pour l'interpréter ? C'est le mécanisme d'**attention** : pour construire la représentation d'un mot, le modèle pondère l'importance de tous les autres mots de la séquence, et le fait pour chaque mot en même temps plutôt que les uns après les autres. Prenez la phrase « l'avocat a plaidé au tribunal » : un lecteur humain accorde spontanément plus d'attention à « plaidé » et « tribunal » qu'au reste de la phrase pour comprendre qu'il s'agit du métier, et non du fruit. L'attention fait exactement ce travail, automatiquement, pour chaque mot à la fois, là où un RNN devait lire tous les mots précédents un par un pour espérer garder ce lien en mémoire. Comme l'attention seule ne code pas naturellement l'ordre des mots, une information de position est ajoutée à chaque élément de la séquence.

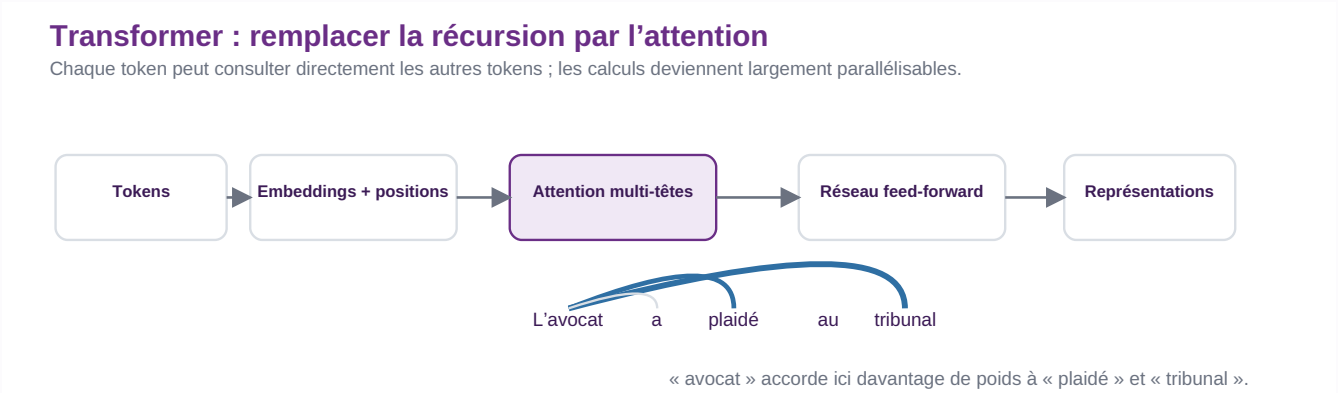


Figure 17. Vue pédagogique d'un Transformer : les tokens sont encodés avec leur position, puis l'attention multi-têtes construit des représentations contextuelles avant les transformations feed-forward. Schéma original inspiré de Vaswani et al. (2017).

SOUS LE CAPOT – lecture avancée facultative

Ce que calcule réellement l'attention. Pour chaque token, le modèle fabrique trois représentations : une **requête** Q (ce que je cherche), une **clé** K (ce que je peux signaler aux autres) et une **valeur** V (l'information que je peux transmettre). La requête d'un mot est comparée aux clés de tous les autres ; ces comparaisons donnent des **scores d'attention**. Les valeurs sont ensuite pondérées par ces scores pour construire une nouvelle représentation du mot, enrichie par son contexte. Plusieurs têtes d'attention effectuent ce calcul en parallèle et peuvent apprendre des relations différentes. Comme ce mécanisme ne connaît pas spontanément l'ordre des mots, on ajoute aussi une information de **position**.

Ce changement résout d'un coup les deux limites des RNN : tous les mots peuvent être traités **en parallèle**, ce qui permet d'entraîner des modèles beaucoup plus grands, beaucoup plus vite, sur beaucoup plus de données, un facteur décisif pour l'essor des grands modèles de langage à partir de 2018–2019, et un mot en fin de texte peut regarder directement un mot en tout début de texte, sans passer par une longue chaîne d'étapes intermédiaires où l'information risquait de s'atténuer. Il faut cependant se garder d'un raccourci : le Transformer ne « comprend » pas le contexte au sens humain du terme. Il apprend des représentations contextuelles en exploitant les relations statistiques entre les éléments d'une séquence, un peu comme on peut deviner le sens d'un mot inconnu en observant les mots qui l'entourent, sans avoir de véritable compréhension du monde.

Cette idée devient particulièrement concrète avec **BERT**, proposé par Devlin et ses collègues en 2018 et publié en 2019[23]. Pendant son pré-entraînement, certains mots sont masqués et le modèle doit les reconstruire. Comme BERT est bidirectionnel, le mot manquant peut mobiliser simultanément ce qui le précède et ce qui le suit, au lieu de ne regarder que vers la gauche.

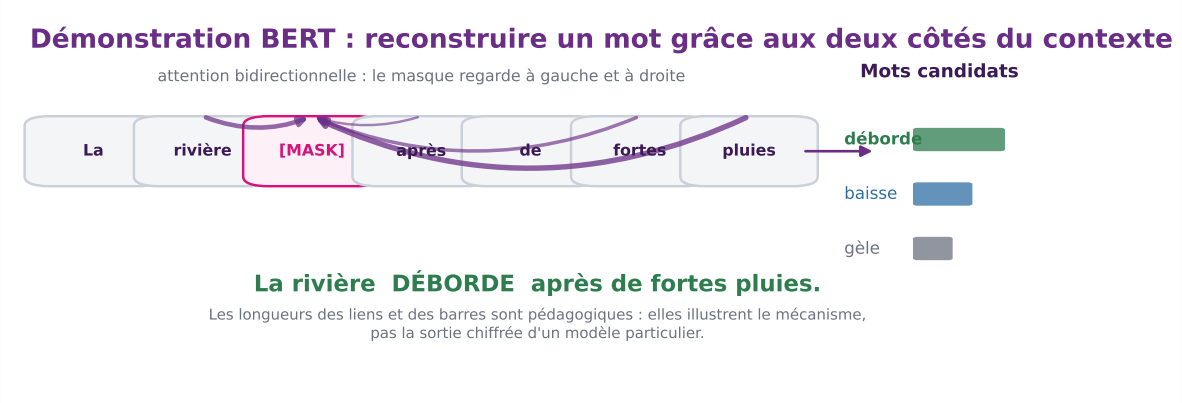


Figure 18. Démonstration du pré-entraînement masqué de BERT. Pour compléter « La rivière [MASK] après de fortes pluies », le token masqué mobilise le contexte situé des deux côtés et privilégie « déborde ». Les épaisseurs et les barres illustrent le mécanisme sans prétendre reproduire les scores d'un modèle particulier. Redessin original d'après Devlin et al. (2019)[23].

Domaine	Application des Transformers
Médecine	Synthèse de dossiers patients, aide à la rédaction de comptes rendus, extraction d'informations cliniques depuis du texte libre.
Agriculture	Chatbots de conseil agronomique, traduction et diffusion de connaissances entre langues et régions.
Industrie	Analyse de rapports d'incidents et de manuels techniques, assistants conversationnels pour techniciens de maintenance.
Traduction	Les moteurs modernes (Google Traduction, DeepL) reposent aujourd'hui sur des Transformers, avec une qualité largement supérieure aux approches statistiques ou aux RNN.

Table 4. Cas d'usage des architectures Transformer

2018–2026 : DU TRANSFORMER AUX GRANDS MODÈLES DE LANGAGE

Le Transformer n'est pas encore, à lui seul, un assistant conversationnel. Le changement d'échelle vient ensuite : pré-entraînement sur de très grands corpus, adaptation à de nombreuses tâches, augmentation du nombre de paramètres et du volume de calcul. BERT (2018) popularise le pré-entraînement bidirectionnel pour comprendre le texte ; la famille GPT pousse l'approche générative autorégressive ; à partir de 2022, l'interface conversationnelle rend ces modèles visibles du grand public. Dans les années suivantes, texte, image, audio et vidéo convergent progressivement vers des systèmes **multimodaux**.

Ce point est essentiel pour lire correctement l'histoire récente : le **LLM n'est pas une architecture entièrement nouvelle qui remplace le Transformer**. C'est, dans la plupart des cas, un Transformer entraîné à très grande échelle, avec une stratégie d'apprentissage, d'alignement et d'utilisation qui a elle aussi fortement évolué. La rupture visible pour l'utilisateur vient donc autant de l'**échelle** et des **données** que de l'architecture elle-même.

2012–2019 : la décennie des architectures décisives

Entre 2012 et la fin des années 2010, plusieurs ruptures successives transforment le domaine de la vision et de la génération : AlexNet popularise l'apprentissage profond, les GAN ouvrent une nouvelle voie pour l'IA générative, et ResNet permet d'entraîner des réseaux beaucoup plus profonds qu'auparavant. Elles préparent directement l'essor actuel de l'IA générative.

Le duel du faussaire et de l'expert : les GAN

Jusque-là, les réseaux de neurones savaient surtout **reconnaître** ou **classer**, « cette image contient un chat ». Mais pouvait-on entraîner un réseau à **créer** de nouvelles images réalistes, plutôt qu'à se contenter de les analyser ? Le défi est que personne ne peut fournir de « bonne réponse » toute faite pour juger si une image générée est réussie. En 2014, Ian Goodfellow propose[11] une solution aussi simple qu'ingénieuse : faire juger cette qualité par un second réseau, entraîné en parallèle du premier. C'est la naissance des **GAN** (*Generative Adversarial Networks*).

Le principe est celui d'un duel. Un **générateur** produit de faux exemples à partir d'un signal aléatoire ; un **discriminateur** tente de distinguer les vrais exemples des faux. C'est exactement le jeu du faussaire de tableaux face à l'expert en authentification. Au début, le faussaire est maladroit et l'expert le démasque facilement. Mais à force d'échecs, le faussaire s'améliore ; l'expert doit alors affiner son œil pour continuer à le repérer. Ce duel, répété des milliers de fois, pousse le faussaire, le générateur, à produire des faux de plus en plus convaincants, jusqu'à des images de synthèse quasiment indiscernables de vraies photographies.

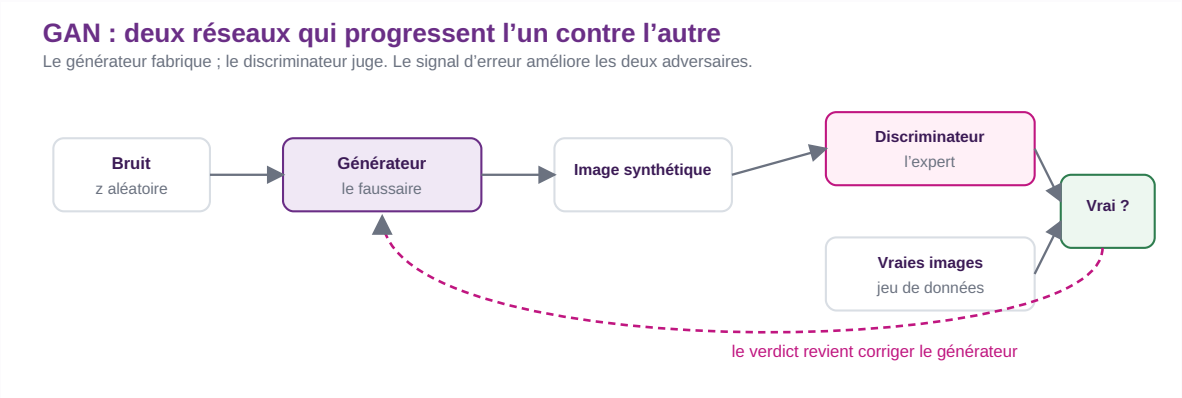


Figure 19. Architecture d'un GAN : un générateur apprend à tromper un discriminateur, qui apprend simultanément à distinguer données réelles et données synthétiques.

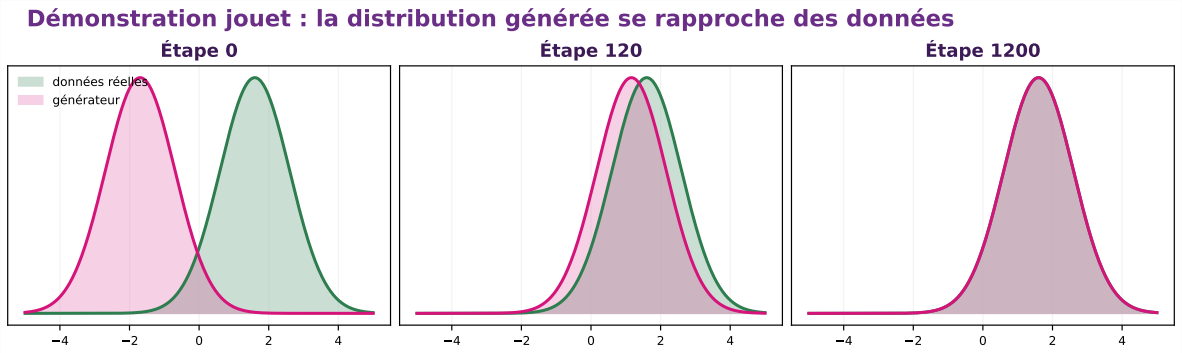


Figure 20. Démonstration jouet d'un apprentissage adversarial. Dans ce GAN unidimensionnel minimal, le générateur déplace progressivement sa distribution vers celle des données réelles, sous le signal fourni par le discriminateur. Illustration reproductible, volontairement simplifiée (graine 21), d'après Goodfellow et al. (2014)[11] ; code : `scripts/generate_demo_figures.py`.

SOUS LE CAPOT – lecture avancée facultative

Deux réseaux qui progressent ensemble. Le **générateur** transforme un vecteur aléatoire en un exemple synthétique. Le **discriminateur** reçoit à la fois des données réelles et des données générées et apprend à dire lesquelles sont authentiques. L'erreur du discriminateur fournit au générateur un signal pour s'améliorer. Aucun des deux réseaux ne possède donc, seul, la solution : c'est leur compétition qui produit l'apprentissage. L'analogie du faussaire et de l'expert résume bien ce jeu, mais son équilibre est délicat, ce qui explique aussi l'instabilité historique des GAN.

En médecine, les GAN permettent d'augmenter des jeux de données rares, générer des images synthétiques mais réalistes de tumeurs peu fréquentes pour mieux entraîner des outils de diagnostic, ou d'améliorer la résolution d'images d'IRM. En agriculture, ils génèrent des données synthétiques de maladies de plantes peu représentées dans les bases existantes. Dans l'industrie, ils simulent des défauts rares (fissures, corrosion) pour entraîner des systèmes de contrôle qualité même quand peu d'exemples réels sont disponibles.

Ne plus tout réapprendre à chaque étage : ResNet

En empilant toujours plus de couches, les chercheurs espéraient des réseaux toujours plus performants. Mais au-delà d'un certain nombre de couches, les performances se dégradaient au contraire, y compris sur les données d'entraînement elles-mêmes. Le signal d'erreur, en traversant trop de couches lors de la rétropropagation, finissait par s'atténuer ou se déformer avant d'atteindre les premières couches du réseau, un peu comme une consigne qui se perd en traversant trop d'intermédiaires. En 2015, He et ses collègues proposent **ResNet**[12] et une idée simple pour contourner le problème : au lieu de demander à chaque bloc d'apprendre une transformation complète depuis zéro, on lui demande d'apprendre seulement une **correction** par rapport à son entrée, via une connexion courte (*skip connection*) qui laisse l'information circuler directement. C'est ce que fait un éditeur qui, plutôt que de réécrire entièrement un document à chaque relecture, transmet le texte précédent presque tel quel et n'ajoute que des corrections en marge : chaque relecteur n'a besoin d'apprendre que ce qui doit changer, pas de tout reconstruire. ResNet devient ainsi la référence en vision par ordinateur, et ouvre la voie à des applications demandant une finesse d'analyse accrue : classification de lésions

cutanées en médecine, identification précise de stades de maturité en agriculture, détection de microfissures dans l'industrie.

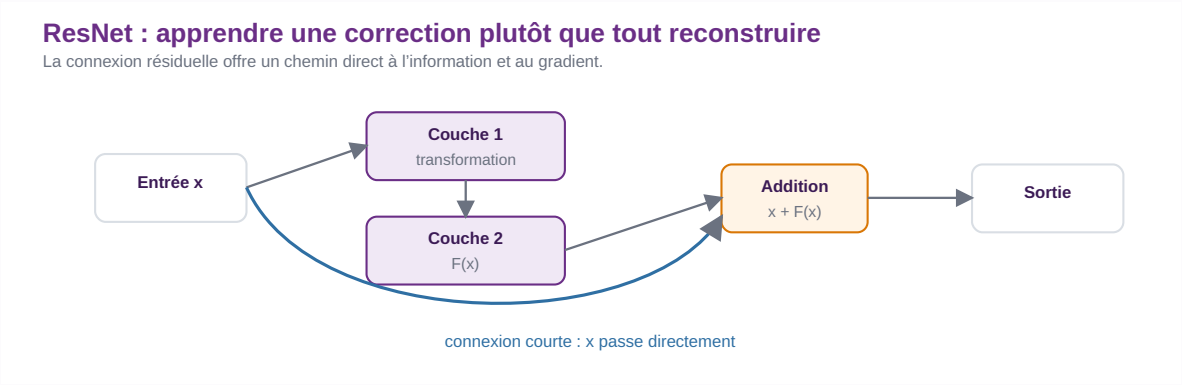


Figure 21. Bloc résiduel : au lieu d'apprendre une transformation complète, le réseau apprend une correction $F(x)$ ajoutée à l'entrée x.

Des GAN aux modèles de diffusion

Les GAN ont constitué une étape majeure de l'IA générative, mais leur duel intégré s'avère en pratique difficile à équilibrer. Si le faussaire prend une longueur d'avance, l'expert n'apprend plus rien ; si l'expert devient trop sévère, le faussaire décroche et cesse de progresser. Cette instabilité, ajoutée à une diversité parfois limitée des images produites, pousse les chercheurs vers une stratégie différente[14][15] : plutôt qu'un duel entre deux réseaux, un seul réseau apprend patiemment à inverser un processus de bruitage progressif. Ce sont les **modèles de diffusion**.

Leur logique s'inspire de la physique statistique. Pendant l'apprentissage, on étudie comment une image peut être progressivement recouverte de bruit, jusqu'à devenir un signal entièrement brouillé, un écran de neige télévisuelle. Puis, lors de la génération, le modèle apprend à parcourir le chemin inverse : il part d'un bruit pur et le **débruite** pas à pas, retirant à chaque étape un peu du désordre initial jusqu'à révéler une image nette et cohérente. C'est le geste d'un sculpteur qui, face à un bloc de neige compacte, enlève un peu de matière à chaque coup de ciseau, guidé par une idée de la forme finale, procédant par retouches successives plutôt qu'en un seul geste. Le générateur d'un GAN, lui, transforme directement un signal en image en une seule passe, comme un peintre qui exécute son tableau d'un trait, sans repentir. Cette différence de méthode explique en grande partie pourquoi les deux familles de modèles se comportent si différemment, et pourquoi la diffusion s'est imposée pour la génération d'images de haute qualité.

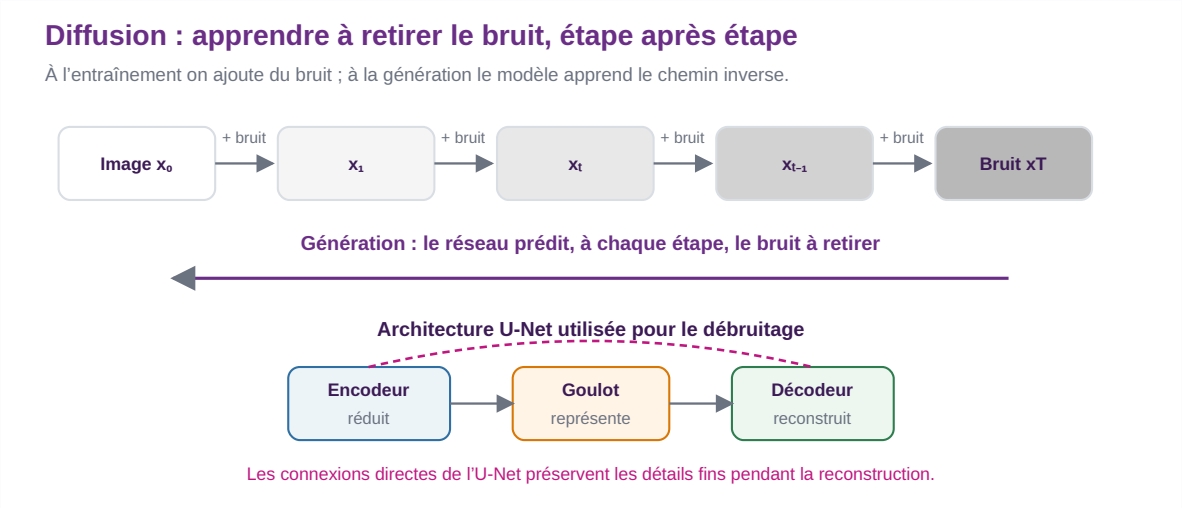


Figure 22. Principe d'un modèle de diffusion : le processus direct ajoute progressivement du bruit ; le modèle apprend le processus inverse de débruitage. L'encart rappelle l'architecture U-Net fréquemment utilisée pour prédire le bruit.

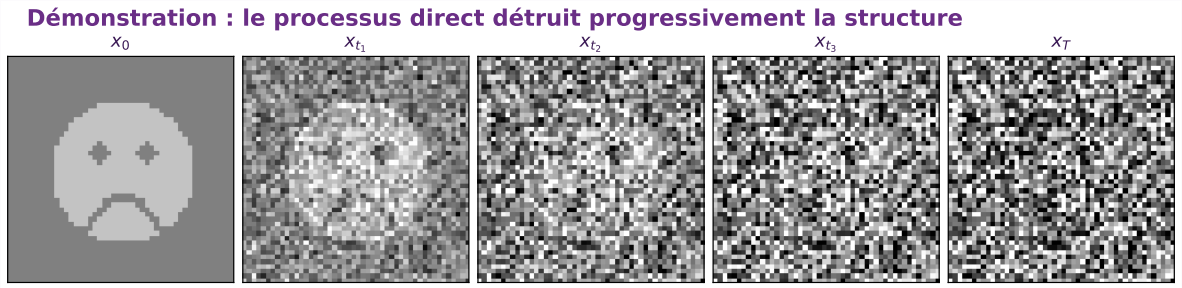


Figure 23. Démonstration du processus direct de diffusion. Une image synthétique est mélangée au même bruit gaussien selon cinq niveaux : sa structure disparaît progressivement jusqu'à x_T. Le réseau de diffusion apprend à parcourir le chemin inverse représenté dans le schéma précédent. Illustration originale reproductible (graine 33) d'après Ho, Jain et Abbeel (2020)[15] ; code : `scripts/generate_demo_figures.py`.

SOUS LE CAPOT – lecture avancée facultative

Générer, c'est apprendre à débruiter. Pendant l'entraînement, on part d'images réelles auxquelles on ajoute progressivement du bruit. Le réseau apprend à estimer, étape après étape, la composante de bruit qu'il faudrait retirer. À la génération, on inverse le parcours : on part d'un bruit aléatoire et on applique plusieurs corrections successives jusqu'à obtenir une image cohérente. Dans les modèles texte-image, le texte fournit une **condition** qui oriente ces corrections vers le contenu demandé. Contrairement au GAN, l'image n'apparaît donc pas en un seul passage ; elle se construit par raffinements successifs.

Les jalons de cette évolution sont resserrés dans le temps : après les GAN en 2014, les modèles probabilistes de diffusion connaissent une accélération importante en 2020, deviennent largement visibles auprès du grand public en 2022 avec les systèmes de génération d'images à partir de texte, avant que la génération ne devienne véritablement multimodale dans les années 2020, combinant texte, image, audio et vidéo. En médecine, la diffusion sert à débruiter des images accélérées (IRM rapides, scanners à faible dose) et à générer des données synthétiques

préservant la confidentialité des patients réels. En agriculture, elle produit des images synthétiques de cultures dans des conditions variées, sécheresse, maladies, stades de croissance, pour enrichir les jeux de données des outils de diagnostic visuel. Dans l'industrie, elle accélère la génération de variantes visuelles de produits en phase de conception, ou simule des scénarios d'usure pour anticiper la maintenance.

Ce que cette histoire nous apprend

L'histoire de l'IA n'est pas une succession de miracles indépendants : chaque rupture s'appuie sur une idée plus ancienne, devenue réellement efficace grâce à une nouvelle architecture, davantage de données et davantage de puissance de calcul. Et chaque technologie naît pour répondre à un problème concret, avant de buter à son tour sur ses propres limites, ces limites devenant la raison d'être de la technologie suivante : les LSTM répondent à une faiblesse des RNN, le Transformer répond aux limites des LSTM, ResNet répond à une limite des réseaux trop profonds, la diffusion répond à l'instabilité des GAN.

Trois autres leçons se dégagent. D'abord, une bonne idée peut précéder de plusieurs décennies les moyens techniques nécessaires pour l'exploiter : le neurone artificiel de 1943 n'a vraiment décollé que dans les années 2010. Ensuite, les progrès viennent presque toujours de la combinaison entre algorithmes, données et calcul, comme une recette qui échoue si un seul ingrédient manque. Enfin, les technologies plus anciennes, systèmes experts, logique floue, neuro-flou, n'ont pas disparu : elles continuent de fonctionner discrètement là où l'explicabilité ou la sobriété comptent plus que la performance brute, un rappel utile à l'heure des grands modèles de langage.

Et la suite ? De l'IA qui répond à l'IA qui agit

L'IA générative a d'abord impressionné par sa capacité à **produire** du texte, des images, du son ou du code. La rupture suivante est moins spectaculaire à regarder, mais probablement plus profonde dans les usages : ces modèles deviennent progressivement capables de **s'insérer dans une chaîne d'action**. Un modèle de langage peut déjà chercher une information dans une base documentaire, appeler une API, interroger une base de données, exécuter du code, remplir un formulaire ou transmettre le résultat à un autre outil. On parle alors d'**agents**. Le mot peut donner l'impression d'une machine autonome qui pense seule ; en pratique, les systèmes les plus robustes ressemblent plutôt à des orchestrateurs : le modèle analyse la situation, choisit une prochaine étape parmi les actions disponibles, observe le résultat, puis décide de la suite. Cette succession *observer, choisir, agir, vérifier* produit une forme de **pseudo-raisonnement** très utile, sans qu'il soit nécessaire de lui attribuer un raisonnement humain.

CE QUI FONCTIONNE DÉJÀ BIEN

En 2026, les usages les plus mûrs ne sont pas les agents totalement autonomes, mais les **tâches bien délimitées et vérifiables**. Résumer et comparer des documents, extraire des informations, rechercher dans une base de connaissances, produire ou modifier du code,

préparer un rapport, appeler des outils métier ou enchaîner quelques opérations sont déjà des cas d'usage solides lorsque le système dispose du bon contexte, d'outils clairement définis et de contrôles. Les retours de production montrent d'ailleurs des agents encore relativement courts : dans une étude menée en 2026 auprès de praticiens, 68 % des systèmes observés exécutaient au plus dix étapes avant une intervention humaine[19]. La bonne question n'est donc plus seulement « que sait générer le modèle ? », mais « quelle partie d'un processus peut-on lui déléguer de manière contrôlée ? ».

La frontière se déplace néanmoins rapidement. Les modèles deviennent meilleurs pour décomposer un objectif en sous-problèmes, utiliser plusieurs outils, conserver un contexte plus long, vérifier certains résultats et coordonner plusieurs actions. On peut donc s'attendre à des agents capables de prendre en charge des processus plus longs, par exemple préparer une analyse à partir de plusieurs sources, modifier un logiciel puis lancer ses tests, surveiller un équipement et proposer une action, ou coordonner plusieurs assistants spécialisés. Mais **plus une chaîne est longue, plus une petite erreur peut se propager**. Les évaluations récentes montrent que les progrès en capacité sont plus rapides que les progrès en fiabilité : un agent peut réussir une tâche complexe une fois et échouer lors d'une tentative presque identique[20]. L'enjeu des prochaines années sera donc autant l'**ingénierie de la confiance**, vérification, traçabilité, droits d'accès, mémoire, contrôle humain, que l'augmentation brute des performances des modèles.

POINT DE VIGILANCE

Plus autonome ne veut pas nécessairement dire plus intelligent. Un agent peut donner l'impression de raisonner parce qu'il planifie plusieurs étapes, consulte des sources et corrige certaines erreurs. Pourtant, il reste dépendant de son modèle, de son contexte et des outils qu'on lui fournit. L'autonomie augmente surtout la portée de ses actions ; elle augmente donc aussi les conséquences possibles d'une erreur. Pour les usages sensibles, un système qui sait demander une validation humaine au bon moment peut être préférable à un système cherchant à tout faire seul.

Faire mieux avec moins

Une autre évolution sera peut-être moins visible, mais tout aussi importante : la recherche ne consiste plus uniquement à construire des modèles toujours plus grands. Elle cherche également à obtenir **davantage de performance avec moins de paramètres, moins de données et moins d'énergie**. Petits modèles spécialisés, distillation, quantification, architectures plus efficaces, calcul local sur un téléphone ou une machine industrielle, et routage d'une requête vers le modèle juste assez puissant constituent autant de pistes d'une **IA frugale**. L'enjeu devient encore plus important avec les agents et les modèles dits de raisonnement, car une requête longue peut demander beaucoup plus de calcul qu'une réponse classique. Une étude de 2026 estime ainsi que les requêtes longues de type raisonnement peuvent multiplier d'environ treize fois l'énergie d'inférence par rapport à une requête standard, tout en montrant qu'une combinaison d'améliorations algorithmiques, logicielles et matérielles pourrait réduire fortement ce coût[21]. La prochaine rupture ne sera donc peut-être pas seulement un modèle

plus puissant, mais un système qui sait **quand utiliser un grand modèle, quand utiliser un petit modèle, et quand ne pas utiliser d'IA du tout**.

Enfin, plus l'IA intervient dans une décision ou déclenche une action, plus une vieille question redevient centrale : **pourquoi a-t-elle décidé cela ?** L'explicabilité (XAI) cherche à identifier les informations qui ont influencé une prédiction, à rendre les mécanismes de décision plus inspectables ou à produire des justifications que l'on puisse confronter aux faits. Ici encore, il faut rester prudent : une explication produite en langage naturel par un modèle n'est pas nécessairement l'explication réelle de son fonctionnement. Les recherches actuelles cherchent donc à rapprocher **performance, sobriété, traçabilité et explicabilité**, quatre objectifs longtemps étudiés séparément[22]. Après avoir appris aux machines à calculer, reconnaître, générer puis agir, la prochaine étape pourrait ainsi être de leur apprendre à agir **de façon plus fiable, plus compréhensible et avec moins de ressources**.

> **SOUS LE CAPOT** – lecture avancée facultative
>
> **Repères chronologiques**
>
> - **1943–1956** : les fondations théoriques (neurone artificiel, test de Turing, naissance du terme « IA »).
> - **1960–1990** : deux hivers de l'IA, marqués par les systèmes experts, la logique floue et le neuro-flou.
> - **1990–2010** : le retour progressif de l'apprentissage (LeNet-5, RNN et LSTM, traduction statistique puis neuronale, essor du Web).
> - **2012–2019** : la décennie des grandes architectures (AlexNet, GAN, ResNet, Transformer).
> - **2020–2026** : l'ère générative multimodale, portée par les modèles de diffusion et les grands modèles de langage.

Crédits des illustrations historiques

Les schémas d'architecture sont des **redessins originaux en vectoriel** (SVG/PDF), conçus pour cette version à partir des architectures publiées. Les quatre figures de démonstration consacrées à Hopfield, Kohonen, aux GAN et à la diffusion sont des **simulations originales et reproductibles**, générées par le script fourni dans le projet ; leurs paramètres, graines aléatoires et articles de référence sont précisés dans les légendes. Les photographies historiques ont été intégrées pour donner un visage aux ruptures scientifiques et techniques. Les crédits sont rappelés sous chaque image : portrait d'Alan Turing, Elliott & Fry, domaine public ; Dartmouth Hall, Daderot, CC0 1.0 ; Mark I Perceptron, National Museum of the U.S. Navy, domaine public ; Teuvo Kohonen, Teknillinen korkeakoulu / Aalto University, CC BY 4.0 ; IBM Deep Blue au Computer History Museum, Anton Chiang, CC BY 2.0. La photographie du match Kasparov–Deep Blue provient d'une archive IBM fournie avec le projet.

Références

1. McCulloch, W. S. & Pitts, W. (1943). *A Logical Calculus of the Ideas Immanent in Nervous Activity*. Bulletin of Mathematical Biophysics, 5, 115–133.
2. Turing, A. M. (1950). *Computing Machinery and Intelligence*. Mind, 59(236), 433–460.
3. Zadeh, L. A. (1965). *Fuzzy Sets*. Information and Control, 8(3), 338–353.

4. Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier/North-Holland.

5. McDermott, J. (1982). *R1: A Rule-Based Configurer of Computer Systems* (XCON). Artificial Intelligence, 19(1), 39–88.

6. Rosenblatt, F. (1958). *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. Psychological Review, 65(6), 386–408.

7. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). *Learning Representations by Back-Propagating Errors*. Nature, 323, 533–536.

8. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition* (LeNet-5). Proceedings of the IEEE, 86(11), 2278–2324.

9. Hochreiter, S. & Schmidhuber, J. (1997). *Long Short-Term Memory*. Neural Computation, 9(8), 1735–1780.

10. Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks* (AlexNet). Advances in Neural Information Processing Systems, 25.

11. Goodfellow, I., Pouget-Abadie, J., Mirza, M. et al. (2014). *Generative Adversarial Networks*. Advances in Neural Information Processing Systems, 27.

12. He, K., Zhang, X., Ren, S. & Sun, J. (2016). *Deep Residual Learning for Image Recognition* (ResNet). Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.

13. Vaswani, A., Shazeer, N., Parmar, N. et al. (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems, 30, 5998–6008.

14. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. & Ganguli, S. (2015). *Deep Unsupervised Learning Using Nonequilibrium Thermodynamics*. Proceedings of the 32nd International Conference on Machine Learning (ICML), 2256–2265.

15. Ho, J., Jain, A. & Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models*. Advances in Neural Information Processing Systems, 33, 6840–6851.

16. Hopfield, J. J. (1982). *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*. Proceedings of the National Academy of Sciences, 79(8), 2554–2558.

17. Kohonen, T. (1982). *Self-Organized Formation of Topologically Correct Feature Maps*. Biological Cybernetics, 43, 59–69.

18. Campbell, M., Hoane, A. J. & Hsu, F. (2002). *Deep Blue*. Artificial Intelligence, 134(1–2), 57–83.

19. Rabanser, S., Kapoor, S., Kirgis, P., Liu, K., Utpala, S. & Narayanan, A. (2026). *Measuring Agents in Production*. ICLR 2026.

20. Rabanser, S., Kapoor, S., Kirgis, P., Liu, K., Utpala, S. & Narayanan, A. (2026). *Towards a Science of AI Agent Reliability*. arXiv:2602.16666.

21. Luccioni, A. S. et al. (2026). *Energy Use of AI Inference, Efficiency Pathways, and Test-Time Scaling*. Joule, 102430.

22. Chouksey, A. et al. (2026). *Frugal and Explainable Artificial Intelligence for Microgrid Energy Management Systems: A Bibliometric View*. Energy Informatics, 9, 61.

23. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proceedings of NAACL-HLT, 4171–4186.