MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université Paris Cité

Evolution des outils IA pour la programmation

13 juin 2025 – David Janiszek

Introduction

↓ Évolution, révolution ou mirage ?

↓ D'où vient l'IAG ?

↓ Le grand bouleversement

↓ Le problème de l'évaluation

D'où vient l'IAG ?

Une (très) brève histoire …

A l'origine …

↓ Les modèles utilisés en IA/ML nécessitent bcp de données (de plus en plus)

↓ Problème : que faire lorsque les données disponibles sont insuffisantes pour progresser ?

↓ Solution : on enrichit les *datasets* avec des données synthétiques !

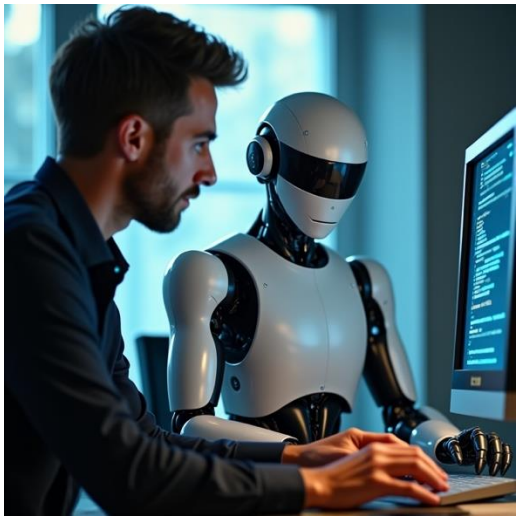
- En pratique : on « inverse » l'architecture
  - reconnaissance ↔ génération
  - Image → mots ↔ mots → image

Evolution du paradigme

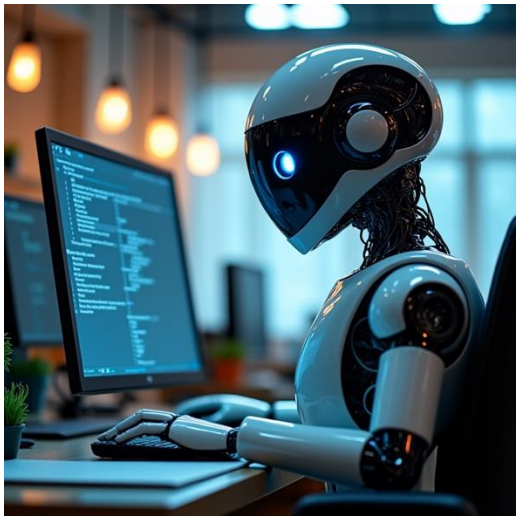
Le grand bouleversement

Il y a 5 ans

Aujourd'hui



Demain ?



Images générées avec flux.dev

Le paradigme de l'assistance (1996-2021)

↓ Système comme assistant (modalité : aide ponctuelle)
- Pas d'IA

↓ Auto-complétion, correction syntaxique, suggestion de variables, détection d'erreurs basiques
- Ex :IntelliSense (Microsoft Visual Studio)

↓ Limitations : suggestions inadaptées, contexte limité, pas de compréhension sémantique

↓ Évaluation simpliste :
- Précision (correct ?)
- Rappel (exhaustif ?)

Le paradigme de génératif (2020-2024)

↓ IA comme co-développeur (modalité : coopération)

↓ Squelette, tests, doc, … (tâches fastidieuses et répétitives)

↓ Productivité : > +30%

↓ Limites:

- Code qui semble correct mais qui ne l'est pas
- Bugs subtils / Failles de sécurité
- Décalage : attentes/promesses vs réalité (benchmarks trompeurs)

Quelques statistiques

↓ Adoption:

- 76 % des développeurs utilisent ou prévoient d'utiliser l'IA (Stack Overflow 2024) versus 70% en 2023

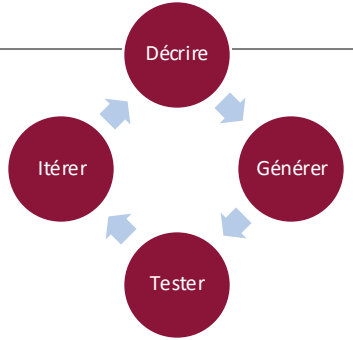
↓ Productivité: approx. +30%

- - 59% de recherches sur internet en lien avec la programmation
- - 48 % de tps sur les tâches complexes
- - 63% de tps sur les tâches simples (How Github copilot boosted developper productivity - Ucsd)

↓ Confiance:

- 72% favorables au outils IA en 2024 versus 77% en 2023 (Stack Overflow 2024)

Vibe-coding



↓ « Il y a une nouvelle façon de coder que j'appelle *vibe coding*, où l'on se laisse porter par les vibes, on embrasse les exponentielles et on oublie que le code existe. » — Andrej Karpathy, février 2025

↓ Définition:

- Le *vibe coding* est une approche de la programmation où l'on décrit un problème en langage naturel à une intelligence artificielle (IA), qui génère ensuite le code correspondant.

↓ Le rôle du développeur évolue : il guide l'IA, teste le code généré et l'affine, plutôt que d'écrire manuellement chaque ligne

↓ Limites/Risques:

- **Compréhension limitée** : Le développeur peut ne pas saisir entièrement le code généré.
- **Difficultés de maintenance** : Le code produit peut être difficile à maintenir ou à adapter.
- **Risque de bugs** : L'IA peut introduire des erreurs subtiles ou des vulnérabilités.

Le paradigme émergent (2024-)

↓ IA comme architecte (modalité : délégation)
- Evolutions : multi-agents, raisonnement, …
- ex: Cursor, Devin AI,…

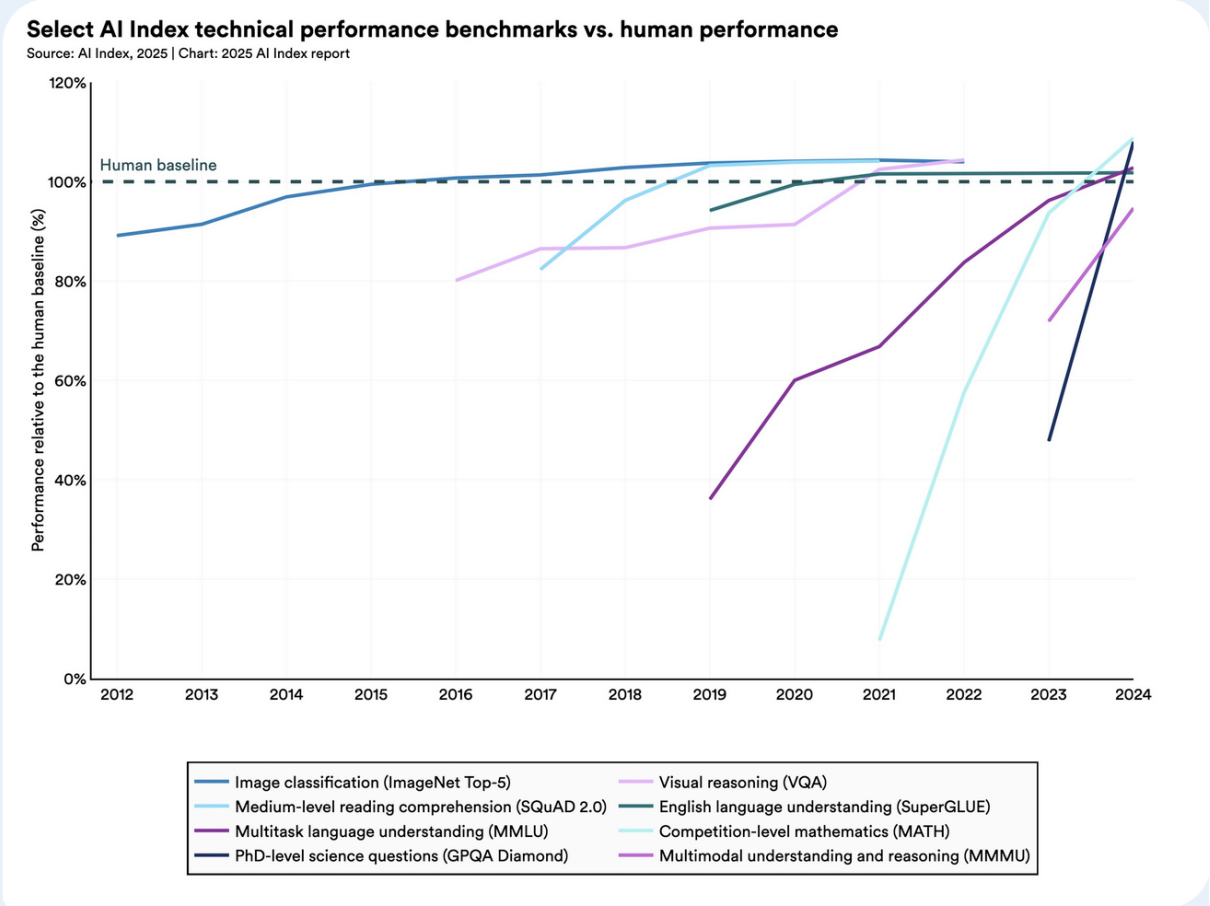
↓ Génération complète de projets

↓ Limites/risques:
- Maintenabilité du code
- Complexité limitée
- Supervision humaine nécessaire (besoins implicites)
- Cout computationnel

↓ Quelle est la limite de la complexité gérable ?

Evaluation des performances

La réalité derrière les métriques…


Select AI Index technical performance benchmarks vs. human performance
Source: AI Index, 2025 | Chart: 2025 AI Index report
Performance relative to the human baseline (%)
120%
100%
80%
60%
40%
20%
0%
Human baseline
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
Image classification (ImageNet Top-5)
Medium-level reading comprehension (SQuAD 2.0)
Multitask language understanding (MMLU)
PhD-level science questions (GPQA Diamond)
Visual reasoning (VQA)
English language understanding (SuperGLUE)
Competition-level mathematics (MATH)
Multimodal understanding and reasoning (MMMU)

Le problème de l'évaluation

↓ HumanEval :
- 164 pb
- niveau : test d'embauche simple

↓ MBPP :
- 1000 pb
- niveau : programmeur débutant

Problèmes simples

↓ Pass@k
- Validation de tests unitaires

↓ Ne prend pas en compte :
- Lisibilité/maintenabilité du code
- Performances du code
- Dette technique générée
- Homogénéité/cohérence

Vers de meilleurs benchmarks ?

↓ SWE-Bench
- 2294 problèmes réels avec solutions issus de 12 projets hébergés sur Github

↓ LiveCodeBench
- 300+ pb (en évolution)
- Eviter la contamination

↓ DS-1000
- 100+ pb réels de datascience (légèrement modifiés pour éviter la contamination)

↓ APPS
- 10000 pb de programmation compétitive / 232,444 solutions

Les biais

↓ Qualité des données d'apprentissage
- StackOverflow, Github, … : répétition du code, bugs persistants, obsolescence, …

↓ Biais scientifique : contamination, auto-contamination
- Données de tests dans l'apprentissage des modèles récents !
- Données synthétiques (biaisées) dans l'apprentissage

↓ Biais linguistique : bons résultats en python (langage le plus utilisé)

↓ Biais culturel : prépondérance de l'anglais et des pratiques américaines

↓ Biais académique : exercices versus problèmes réels

Les défis de l'évaluation

↓ Constat : comparaison par rapport à du code pré-existant

↓ Réalité : si on a besoin d'écrire un programme c'est qu'il n'existe pas !

↓ Comment évaluer une architecture ?

↓ Comment évaluer l'impact à long terme

- Performances, qualité, maintenabilité, coûts d'exploitation, …

↓ Peut-on évaluer la pertinence par rapport au contexte industriel ?

↓ Peut-on mesurer la créativité ?

↓ Comment sortir d'une évaluation en silos ? (il y a une infinité de silos …)

Evolution en cours

Ce qui change vraiment

↓ Démocratisation du code
- Non-développeurs qui créent des programmes utiles

↓ Accélération du prototypage
- De l'idée au MVP en quelques heures

↓ Attention portée sur l'objectif, sur l'architecture, sur la qualité
- Moins de temps sur l'implémentation, plus sur la conception

↓ Augmentation de la productivité
- x3, x5, x10 selon les cas
- Plus de temps pour la collaboration, les échanges, …

Les nouveaux problèmes

↓ Développeurs qui ne savent plus déboguer sans IA

- Panne de Copilot -> équipe bloquée

↓ Baisse de la qualité

- Augmentation des bugs : +41% (paradoxe: svt capable de les identifier mais 1ère version buguée) (Uplevel 2024)
- Explosion de la dette technique (le coût à long terme de solutions rapides ou temporaires)
- Failles de sécurité générées automatiquement
- Code sous-optimal accepté par facilité

Enjeux sociétaux

↓ Emplois et compétences
- Quels développeurs dans 10 ans ? Evolution des profils

↓ Souveraineté technologique
- Dépendance critique aux modèles américains

↓ Impact environnemental
- Augmentation du cout computationnel pour des tâches de niveau constant

↓ Propriété intellectuelle
- Code généré à partir de code sous licence

Conclusion

Constats

↓ L'IA transforme déjà le développement

↓ Les gains de productivité sont importants sur certaines tâches

↓ Les modèles s'améliorent rapidement

↓ L'adoption continue de croitre malgré les déceptions et les risques

Les IAG rendent « bêtes » …

↓ « The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers » (03/2025)

↓ Échantillons : 319 professionnels / Déchargement cognitif (GPS, calculatrice, …)

↓ Solution : Stimuler la pensée critique

- Ex :
  - Identifie les arguments principaux de ce texte et propose une critique pour chacun (analyse)
  - Propose plusieurs réponses, en explicitant les avantages et les inconvénients de chaque solution afin quelle puisse choisir (génération)
  - …

Et en recherche ?

↓ 3 articles dans Nature:

↓ Productivité : 1000 chercheurs (sc. matériaux domaine santé / USA) : +44% papiers / +39% brevets
- Insatisfaction au travail : -> 82 %

↓ Reviews : 12% (+/-5) faite (par/avec) ChatGPT

↓ Bibliographie (accélération, prudence sur la qualité, …)

Questions ouvertes

↓ Jusqu'où l'IA peut-elle gérer la complexité logicielle ?
- Un humain : 10k à 100k loc

↓ L'adoption massive engendrera t'elle une érosion des compétences ?
- Apparition de nouvelles compétences ?

↓ Soutenabilité économique ?
- Modèle économique viable avec le coût des inférences ?
- Inférence à 30k$ avec o3-high (Coût de l'IA : Le Modèle o3 d'OpenAI Plus Cher que Prévu)

↓ Vers une augmentation ou un remplacement des développeurs ?

Quelques recommandations

↓ Maitriser les outils sans perdre les fondamentaux

↓ Expérimenter mais garder l'esprit critique

↓ Evaluer la nécessité et l'impact

↓ Utiliser des modèles adaptés (petits, locaux, open-source, …)

↓ Dans votre pratique, l'IA vous augmente-t-elle ou vous remplace-t-elle ?

Remerciements

↓ Les collègues et les étudiants (alimentent mes réflexions)

↓ Claude Sonnet 4 (préparation)
↓ ChatGPT 4o. (retrouver des références bibliographiques)
↓ Athene V2 (organisation)
↓ Flux.dev (génération des illustrations)

↓ Au fait : le cout des remerciements ! (Sam Altman)