DEVOXX FRANCE 2026

LES FANTASTIQUES PETITS MODÈLES ET QUAND LES UTILISER

About me

Pierre Lepagnol, PhD

Data Scientist chez SCIAM

- Travaille sur les SLMs.
- Thèse soutenue le 10 mars 2026
  « Petits modèles génératifs en contexte industriel »

université PARIS-SACLAY

LISN LABORATOIRE INTERDISCIPLINAIRE DES SCIENCES DU NUMÉRIQUE

Dans la suite des talks

Mesurer l'immesurable

Évaluer les systèmes IA générative

LLMs & Hallucinations

Comprendre, mesurer, maîtriser

Nous connaissons tous les LLM

Large Language Model

- Réseaux de neurones entraînés à prédire le prochain token.
- +100B de paramètres voire 1T (mille milliards de paramètres).

Nous connaissons tous les LLM

Large Language Model

- Réseaux de neurones entraînés à prédire le prochain token.
- **+100B** de paramètres voire **1T** (mille milliards de paramètres).

Coût pour un LLM

DeepSeek-V4-Pro : 1.6T $\Longrightarrow \sim 865GB$ (en FP8)
865GB $\Longrightarrow$ 32xH100 (Nvidia $\sim$ \$25k) $\Longrightarrow$ **\$800k** (GPU uniquement)

Nous connaissons tous les LLM

Large Language Model

- Réseaux de neurones entraînés à prédire le prochain token.
- **+100B** de paramètres voire **1T** (mille milliards de paramètres).
- Modèles généralistes : pertinents dans une **large palette de tâches** (résumé, code, Q&A, raisonnement, etc.).

Nous connaissons tous les LLM

Large Language Model

- Réseaux de neurones entraînés à prédire le prochain token.
- **+100B** de paramètres voire **1T** (mille milliards de paramètres).
- Modèles généralistes : pertinents dans une **large palette de tâches** (résumé, code, Q&A, raisonnement, etc.).
- Accès via **API** cloud (OpenAI, Anthropic, Google…)

Nous connaissons tous les LLM

Large Language Model

- Réseaux de neurones entraînés à prédire le prochain token.
- **+100B** de paramètres voire **1T** (mille milliards de paramètres).
- Modèles généralistes : pertinents dans une **large palette de tâches** (résumé, code, Q&A, raisonnement, etc.).
- Accès via **API** cloud (OpenAI, Anthropic, Google…)
- **Coût** à l'usage/au token.

Nous connaissons tous les LLM

Large Language Model

- Réseaux de neurones entraînés à prédire le prochain token.
- **+100B** de paramètres voire **1T** (mille milliards de paramètres).
- Modèles généralistes : pertinents dans une **large palette de tâches** (résumé, code, Q&A, raisonnement, etc.).
- Accès via **API** cloud (OpenAI, Anthropic, Google…)
- **Coût** à l'usage/au token.
- Données envoyées sur les **serveurs distants** (via le prompt)

Nous connaissons tous les LLM

Large Language Model

- Réseaux de neurones entraînés à prédire le prochain token.
- **+100B** de paramètres voire **1T** (mille milliards de paramètres).
- Modèles généralistes : pertinents dans une **large palette de tâches** (résumé, code, Q&A, raisonnement, etc.).
- Accès via **API** cloud (OpenAI, Anthropic, Google…)
- **Coût** à l'usage/au token.
- Données envoyées sur les **serveurs distants** (via le prompt)

Alternative intéressante aux LLMs → Les SLMs

Dans ce talk

Qu'est ce qu'un SLM ?

Choisir un SLM ou un LLM ?

Comment utiliser un SLM ?

Déployer un SLM

QU'EST CE QU'UN SLM ?

Small Language Models

Small Language Models are the Future of Agentic AI

Peter Belcak[1] **Greg Heinrich**[1] **Shizhe Diao**[1] **Yonggan Fu**[1] **Xin Dong**[1]
Saurav Muralidharan[1] **Yingyan Celine Lin**[1,2] **Pavlo Molchanov**[1]
[1]NVIDIA Research [2]Georgia Institute of Technology
agents-research@nvidia.com

Small Language Models

Définition

Un small language model est un **Language Model** qui peut s'installer sur un **appareil grand public courant** et effectuer des inférences avec une latence suffisamment faible pour **être pratique**.

Small Language Models

Définition

Un small language model est un **Language Model** qui peut s'installer sur un **appareil grand public courant** et effectuer des inférences avec une latence suffisamment faible pour **être pratique**.

1. **Language Model** : modèle comme un LLM (auto-régressif* + entraîné avec Next Token Prediction*)

Small Language Models

Définition

Un small language model est un **Language Model** qui peut s'installer sur un **appareil grand public courant** et effectuer des inférences avec une latence suffisamment faible pour **être pratique**.

1. **Language Model** : modèle comme un LLM (auto-régressif* + entraîné avec Next Token Prediction*)
2. **appareil grand public courant** : laptop, smartphone, workstation sans GPU dédié : souvent $<$ de 16 Go de VRAM.

Small Language Models

Définition

Un small language model est un **Language Model** qui peut s'installer sur un **appareil grand public courant** et effectuer des inférences avec une latence suffisamment faible pour **être pratique**.

1. **Language Model** : modèle comme un LLM (auto-régressif* + entraîné avec Next Token Prediction*)
2. **appareil grand public courant** : laptop, smartphone, workstation sans GPU dédié : souvent $<$ de 16 Go de VRAM.
3. inférences avec une **latence pratique** : réponse perçue fluide par un humain
   - **TTFT** (Time To First Token) : délai entre l'envoi du prompt et la réception du premier token (réactivité).
   - **Tokens/s** : débit de génération

Panthéon des SLMs

	Modèle	Date	Taille	MMLU
Microsoft	Phi-4 mini	Fév 2025	3.8B	67.3 %
Google DeepMind	Gemma 4 E4B	Avr 2026	4.5B	69.4 %*
Qwen	Qwen 3 (4B)	Avr 2025	4.0B	73.0 %
NVIDIA	Nemotron-Nano 4B	Mai 2025	4.0B	- †
	Ministral 3B	Oct 2024	3.0B	60.9 %
Meta	Llama 3.2 (3B)	Sep 2024	3.0B	63.4 %
Hugging Face	SmolLM2 (1.7B)	Nov 2024	1.7B	:
	Baguettotron	Nov 2025	0.3B	-

Propriétés qui font un vrai SLM

- Local friendly
  - Privacy : données 100 % locales.
  - Déploiement sur edge (laptop, smartphones, etc.) ou infra légère.
  - Faible Latence.

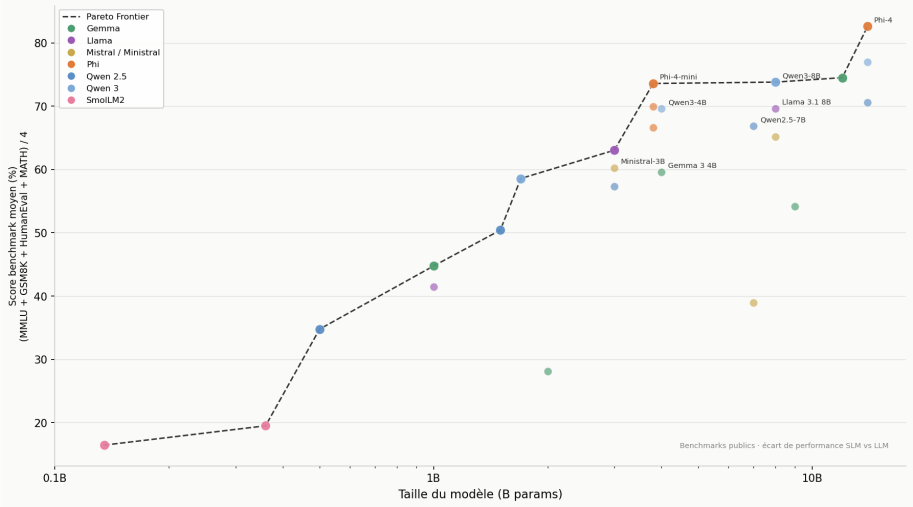
Propriétés qui font un vrai SLM

- Local friendly
  - Privacy : données 100 % locales.
  - Déploiement sur edge (laptop, smartphones, etc.) ou infra légère.
  - Faible Latence.
- Moins cher
  - Matériel plus accessible : $2K (Nvidia 5090) $\rightarrow$ $25K (Nvidia H100)
  - Coût par requête bien inférieur à une API LLM : vraie différence en prod.

Propriétés qui font un vrai SLM

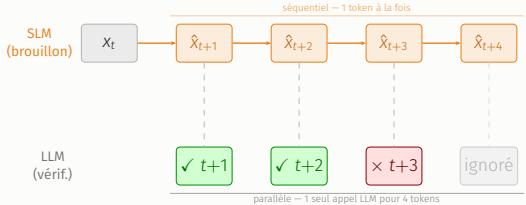
- Local friendly
  - Privacy : données 100 % locales.
  - Déploiement sur edge (laptop, smartphones, etc.) ou infra légère.
  - Faible Latence.
- Moins cher
  - Matériel plus accessible : $2K (Nvidia 5090) → $25K (Nvidia H100)
  - Coût par requête bien inférieur à une API LLM : vraie différence en prod.
- Meilleure plasticité
  - Spécialisation facile via fine-tuning.

Ce n'est pas magique...



À quoi sert un SLM ?

1. Accélérer un LLM



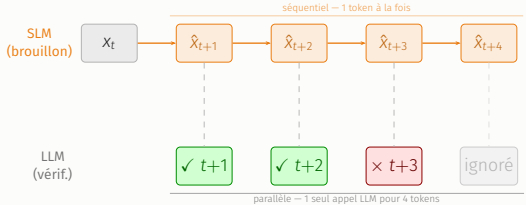
Speculative Decoding :

1. Le SLM produit un brouillon de tokens
2. Le LLM vérifie plusieurs tokens en un seul appel.

Pour les providers : utile pour réduire la latence.

À quoi sert un SLM ?

1. Accélérer un LLM

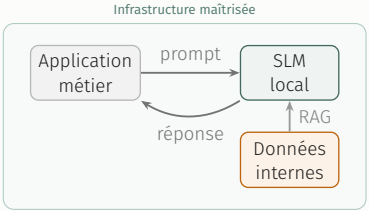


Speculative Decoding :

1. Le SLM produit un brouillon de tokens
2. Le LLM vérifie plusieurs tokens en un seul appel.

Pour les providers : utile pour réduire la latence.

2. Inférence locale / on-prem



- Données sensibles sans aller-retour cloud.
- Latence et coûts plus prévisibles.

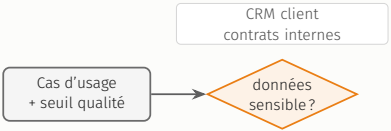
CHOISIR UN SLM OU UN LLM ?

SLM ou LLM : une décision sous contraintes

Cas d'usage
+ seuil qualité

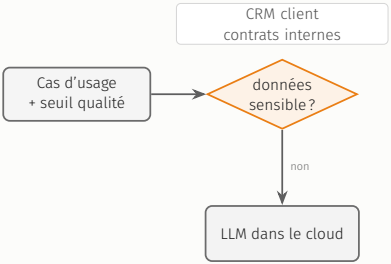
Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

SLM ou LLM : une décision sous contraintes



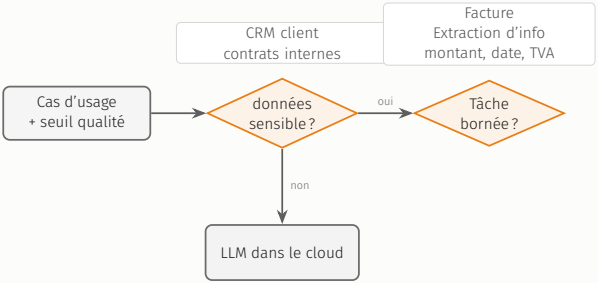
Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

SLM ou LLM : une décision sous contraintes



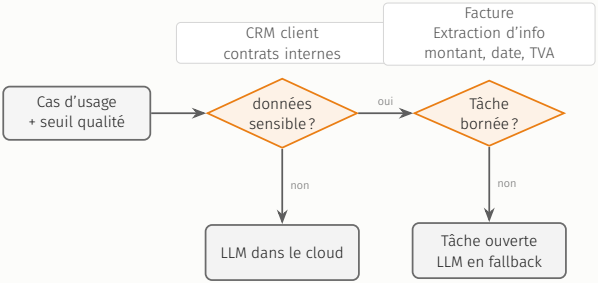
Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

SLM ou LLM : une décision sous contraintes



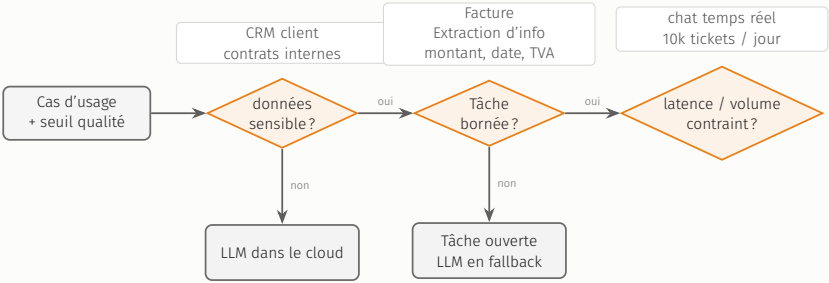
Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

SLM ou LLM : une décision sous contraintes



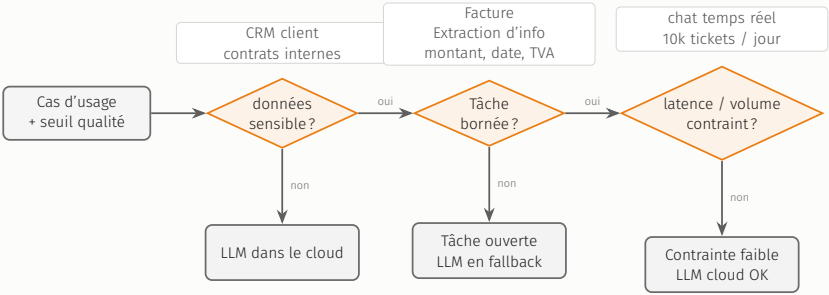
Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

SLM ou LLM : une décision sous contraintes



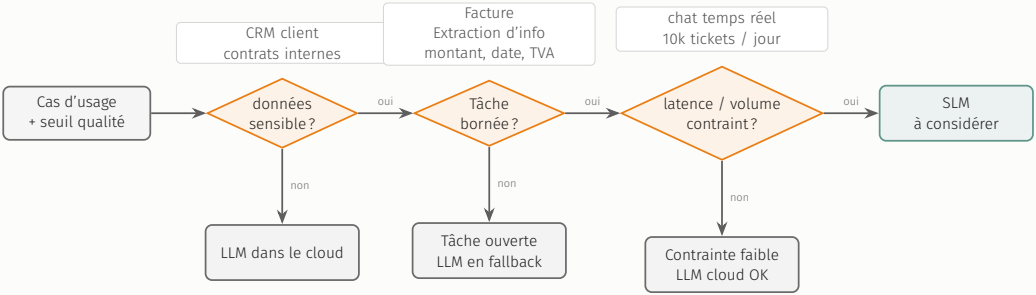
Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

SLM ou LLM : une décision sous contraintes



Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

SLM ou LLM : une décision sous contraintes



Idée clé : choisir le plus petit système qui passe le seuil métier sous contraintes.

Les 4 critères décisionnels

Critère	Signal SLM	Signal LLM
Données	Données sensibles, client, IP, besoin on-prem ou edge.	Données peu sensibles ou contrat cloud acceptable.

Les 4 critères décisionnels

Critère	Signal SLM	Signal LLM
Données	Données sensibles, client, IP, besoin on-prem ou edge.	Données peu sensibles ou contrat cloud acceptable.
Latence	UX interactive, offline, temps réel, réseau instable.	Quelques secondes acceptables, batch, back-office.

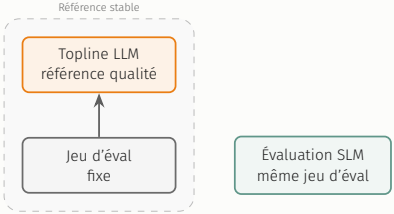
Les 4 critères décisionnels

Critère	Signal SLM	Signal LLM
Données	Données sensibles, client, IP, besoin on-prem ou edge.	Données peu sensibles ou contrat cloud acceptable.
Latence	UX interactive, offline, temps réel, réseau instable.	Quelques secondes acceptables, batch, back-office.
Volume / coût	Usage répétitif, coût/token dominant, besoin de coût prévisible.	Faible volume, coût variable acceptable, POC rapide.

Les 4 critères décisionnels

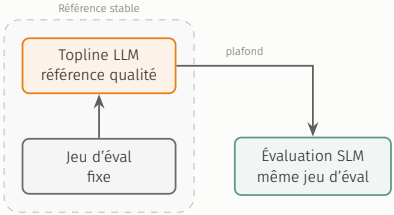
Critère	Signal SLM	Signal LLM
Données	Données sensibles, client, IP, besoin on-prem ou edge.	Données peu sensibles ou contrat cloud acceptable.
Latence	UX interactive, offline, temps réel, réseau instable.	Quelques secondes acceptables, batch, back-office.
Volume / coût	Usage répétitif, coût/token dominant, besoin de coût prévisible.	Faible volume, coût variable acceptable, POC rapide.
Complexité	Tâche bornée : extraction, classification, génération contrainte.	Raisonnement ouvert, multimodal, cas rares et variés.

SLM vs LLM : voir grand, viser petit



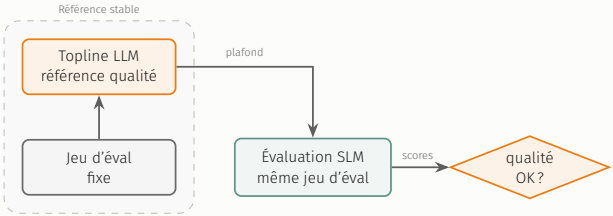
Règle pratique : commencer par le plafond LLM, puis descendre jusqu'au plus petit modèle qui passe qualité, coût et latence.

SLM vs LLM : voir grand, viser petit



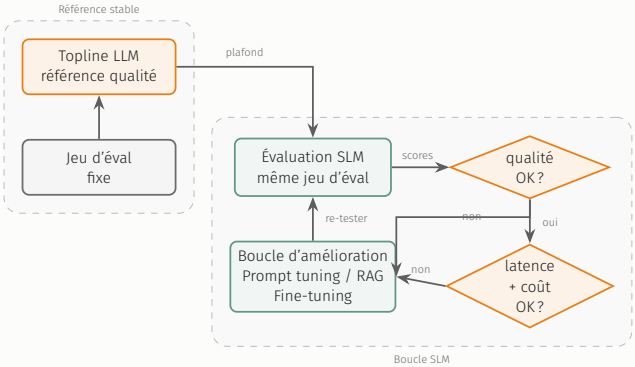
Règle pratique : commencer par le plafond LLM, puis descendre jusqu'au plus petit modèle qui passe qualité, coût et latence.

SLM vs LLM : voir grand, viser petit



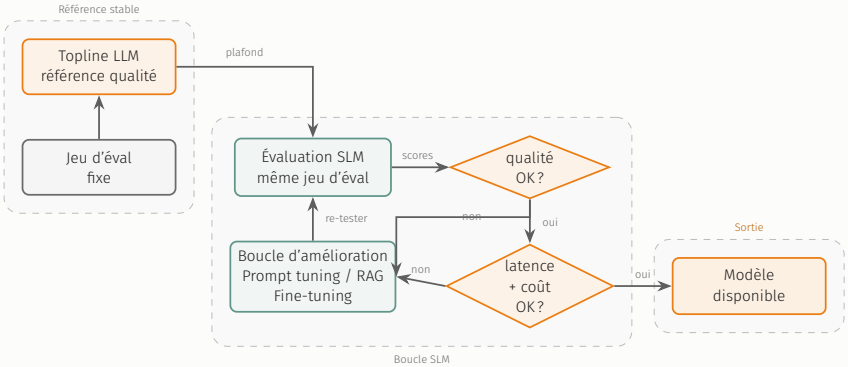
Règle pratique : commencer par le plafond LLM, puis descendre jusqu'au plus petit modèle qui passe qualité, coût et latence.

SLM vs LLM : voir grand, viser petit



Règle pratique : commencer par le plafond LLM, puis descendre jusqu'au plus petit modèle qui passe qualité, coût et latence.

SLM vs LLM : voir grand, viser petit



Règle pratique : commencer par le plafond LLM, puis descendre jusqu'au plus petit modèle qui passe qualité, coût et latence.

Pré-requis & Systèmes finaux

Minimum avant de choisir

1. Données de test.
2. Une topline LLM qui définit le plafond utile.
3. 2 à 3 SLM candidats : petit, moyen, limite hardware.
4. Mesures : qualité, TTFT, tokens/s, coût, erreurs.

Pré-requis & Systèmes finaux

Minimum avant de choisir

1. Données de test.
2. Une topline LLM qui définit le plafond utile.
3. 2 à 3 SLM candidats : petit, moyen, limite hardware.
4. Mesures : qualité, TTFT, tokens/s, coût, erreurs.

Différents systèmes finaux

1. LLM seul
2. SLM seul
3. Hybride : SLM flux courant & LLM Fallback.

Choisir ses candidats SLM : Filtres et Benchmarks

Filtres d'entrée

1. **Contrainte hardware** : VRAM dispo ⇒ taille max du modèle[a].
2. **Taille de contexte** : 128k vs 8k
3. **Support Tool-Calling** : 128k vs 8k
4. **Langue** : le modèle est-il entraîné/évalué en français ? (ex. : CroissantLLM, Baguettotron).
5. **Licence** : commerciale (Apache 2.0, MIT) ou restrictive.[b]

a. quelques astuces possibles : quantisation ou pruning

b. Un modèle "ouvert" (Open Weights) n'est pas forcément "libre" (Open Source).

Choisir ses candidats SLM : Filtres et Benchmarks

Filtres d'entrée

1. **Contrainte hardware** : VRAM dispo ⇒ taille max du modèle[a].
2. **Taille de contexte** : 128k vs 8k
3. **Support Tool-Calling** : 128k vs 8k
4. **Langue** : le modèle est-il entraîné/évalué en français ? (ex. : CroissantLLM, Baguettotron).
5. **Licence** : commerciale (Apache 2.0, MIT) ou restrictive.[b]

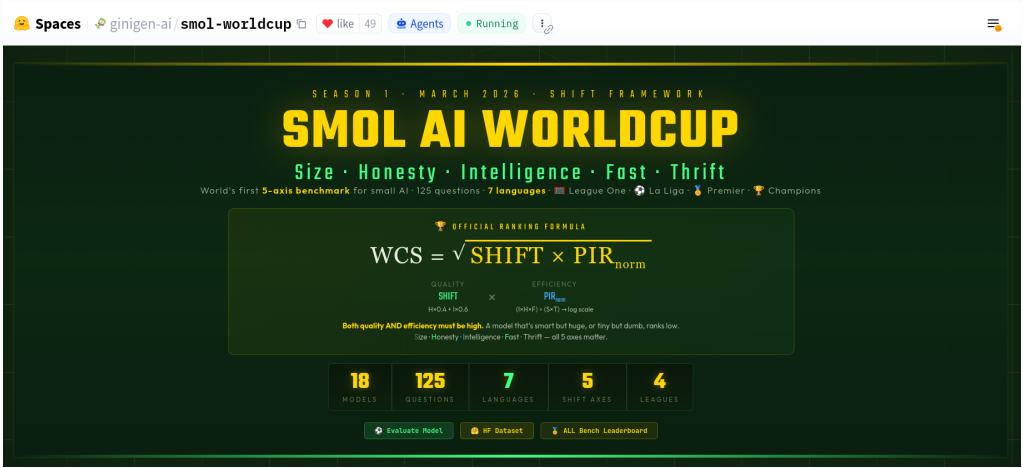
Benchmarks à surveiller

- **MMLU / MMLU-Pro** : Connaissance générale
- **MT-Bench / IFBench** : suivi d'instructions
- **Open LLM Leaderboard / Arena.ai** : Classement à jour
- **SMOL AI Worldcup**
- Own your Benchmark !!

a. quelques astuces possibles : quantisation ou pruning

b. Un modèle "ouvert" (Open Weights) n'est pas forcément "libre" (Open Source).

SMOL AI World Cup



SMOL AI World Cup

#	PLAYER	WCS	PIR	SHIFT	HONESTY	INTEL	UNION	LEAGUE	PARAMS	TOK/S	RAM	TRAP	CALIB	REFUSE	FIX	LOGIC	MATH
1	Gemma-3n-E4B #1 WCS · #1 SHIFT · Top3 Google · PLE	82.2	2136	77.3	82.6	73.8	47.4	La Liga	2B	43.8	2	90	51.5	100	89	68.7	100
2	Llama-4-Scout · Fastest Meta · MoE MoE	79.6	1804.4	74.2	80.9	69.7	47.6	Champions	18	240.5	10	90	41.5	100	92	60	100
3	Qwen3-4B Alibaba · Dense	76.9	858.2	76.8	81.4	73.8	43.2	La Liga	4B	50	2.8	100	42.5	100	83	63.3	100
4	Qwen3-1.7B · Top3 Alibaba · Dense	76.4	2147.5	66.8	71.4	63.7	–	League One	1.7B	30.1	1.2	60	44.5	100	81	53.3	100
5	Qwen3-8B · Honest Alibaba · Dense	73.1	444.9	76.9	87.9	69.6	–	Premier	8B	186.8	5.5	100	65.5	100	86	62	90
6	Darwin-35B-A3B-Opus FINDRAFT · MoE+Hybrid MoE	71.8	358.2	76.8	83.5	72.4	–	Darwin	3B	147.8	18	100	46.5	100	88	75.3	100
7	GPT-OSS-20B OpenAI · MoE MoE	70.2	277.1	76.9	84.9	71.5	54.2	Champions	3.6B	71.9	14	100	63.6	100	76	63.3	100
8	Qwen3.5-35B-A3B Alibaba · MoE MoE	70.2	310.1	75.3	82	70.9	–	Champions	3B	108.7	20	100	43.9	100	84	72	100

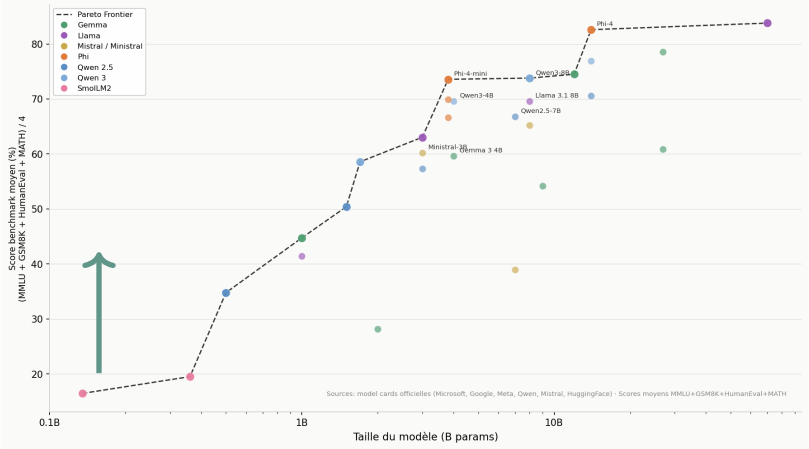
Choisir ses candidats SLM : Sélection

Prenez la dernière version si possible
Shortlist recommandée

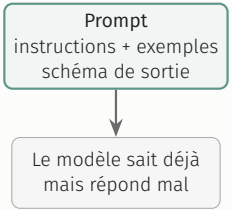
- Gemma 4 (E2B & E4B)
- SmolLM3
- Qwen3.5
- Nemotron-nano
- Ministral & Ministraux

COMMENT UTILISER UN SLM ?

La frontière de Pareto : où se situe votre use case ?

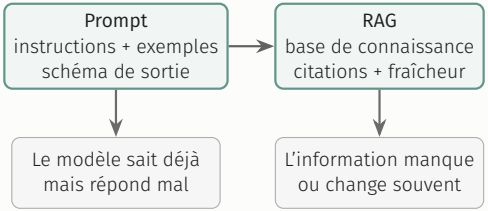


Vanille, RAG, fine-tuning : l'escalier utile (Valide aussi pour LLM)



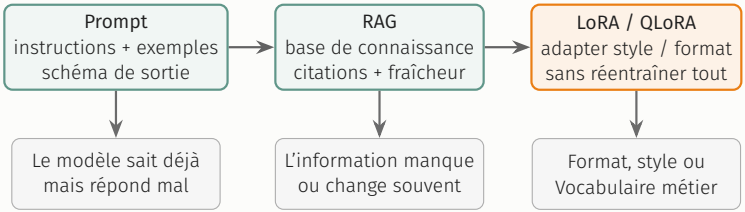
Ne sautez pas de marche : prompt et RAG donnent le diagnostic; LoRA/QLoRA corrige un comportement stable; full fine-tuning reste l'exception.

Vanille, RAG, fine-tuning : l'escalier utile (Valide aussi pour LLM)



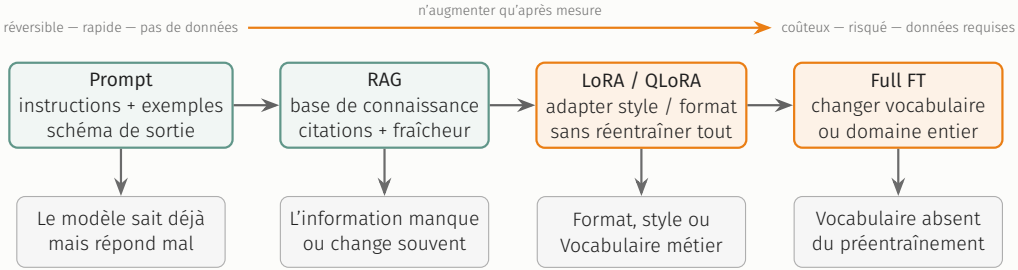
Ne sautez pas de marche : prompt et RAG donnent le diagnostic; LoRA/QLoRA corrige un comportement stable; full fine-tuning reste l'exception.

Vanille, RAG, fine-tuning : l'escalier utile (Valide aussi pour LLM)



Ne sautez pas de marche : prompt et RAG donnent le diagnostic; LoRA/QLoRA corrige un comportement stable; full fine-tuning reste l'exception.

Vanille, RAG, fine-tuning : l'escalier utile (Valide aussi pour LLM)



Ne sautez pas de marche : prompt et RAG donnent le diagnostic; LoRA/QLoRA corrige un comportement stable; full fine-tuning reste l'exception.

Prompting : Plusieurs tests possibles

- Zero-shot : voir les erreurs naturelles.
- Few-shot : 3 à 5 exemples représentatifs.
- Structured Output : valider automatiquement le format.

RAG ou fine-tuning ? Le diagnostic

Erreur observée	Corriger avec RAG	Corriger avec fine-tuning
Faits absents ou courte DLC	Récupérer la bonne source au moment de l'appel	Le modèle oublierait dès que la donnée change

RAG ou fine-tuning ? Le diagnostic

Erreur observée	Corriger avec RAG	Corriger avec fine-tuning
Faits absents ou courte DLC	Récupérer la bonne source au moment de l'appel	Le modèle oublierait dès que la donnée change
Format instable	Aide avec exemples dans le prompt	Corrige le comportement de sortie

RAG ou fine-tuning ? Le diagnostic

Erreur observée	Corriger avec RAG	Corriger avec fine-tuning
Faits absents ou courte DLC	Récupérer la bonne source au moment de l'appel	Le modèle oublierait dès que la donnée change
Format instable	Aide avec exemples dans le prompt	Corrige le comportement de sortie
Ton, style, règles métier	Non (si la règle est permanente)	Oui : apprendre des exemples validés

LoRA, QLoRA : fine-tuner sans se ruiner

LoRA (Low-Rank Adaptation)

Geler les poids originaux, n'entraîner que A et B.

1–5 % des paramètres.

Standard : HuggingFace PEFT, Unsloth, Axolotl.

$\Delta W = B \cdot A \quad (r \ll d)$

QLoRA

Base chargée en 4-bit NF4 (gelée) + adaptateurs bf16.

Fine-tune un 13B sur un GPU 24 Go seulement.

Perte de qualité négligeable vs. full LoRA.

Bibliothèque de training : choisir l'outil au bon niveau

PEFT / TRL

Briques Python : `LoraConfig`, `SFTTrainer`, DPO/GRPO.
Bon choix si vous voulez contrôler le code d'entraînement.

unsloth

Optimise LoRA/QLoRA pour réduire VRAM et temps d'entraînement.
Bon choix sur GPU limité, Colab, workstation.

Axolotl

Pipeline YAML : dataset, entraînement, multi-GPU, export.
Bon choix pour standardiser et industrialiser.

Fine-tuning : les tâches où ça change tout

Très bons candidats

- **Classification et routage** : intention, priorité, tri de tickets.
- **Extraction structurée** : entités, champs, JSON conforme.
- **Génération contrainte** : SQL, snippets, réponses normées.
- **Style métier** : ton, terminologie, refus, disclaimers.

Pré-requis

- 200 à 2 000 exemples propres pour commencer.
- Un jeu d'évaluation fixe avant l'entraînement.
- Une métrique : exact match, validité JSON, score humain.
- CF talk d'Erin

Signal fort : si les mêmes erreurs reviennent après prompt + RAG, l'adapter a probablement une vraie valeur.

DÉPLOYER UN SLM

Déployer : partir de la contrainte, pas de l'outil

Où tourne le modèle ?

1. **Poste dev** : POC local, itération rapide.
2. **Edge** : hors-ligne, CPU/Apple Silicon, faible empreinte.
3. **Serveur interne** : privacy, coût fixe, contrôle total.
4. **Cloud privé** : GPUs managés, montée en charge.

Faire de l'observabilité

- Contexte max et mémoire KV cache.
- QPS/concurrence attendue.
- TTFT, tokens/s, P95 latence.
- Budget VRAM, coût/heure, coût/token.

Moteurs d'inférence : choix concret

Ollama

Démarrer vite, API locale simple, partage de POC sur laptop.

llama.cpp

CPU/edge/Apple Silicon, GGUF, binaire léger, très bon contrôle mémoire.

vLLM

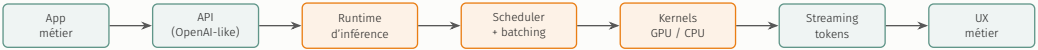
Service GPU : OpenAI-compatible API, continuous batching, KV cache efficace.

TGI / HF

Historiquement très solide ; aujourd'hui surtout utile à connaître pour migration ou plateforme HF.

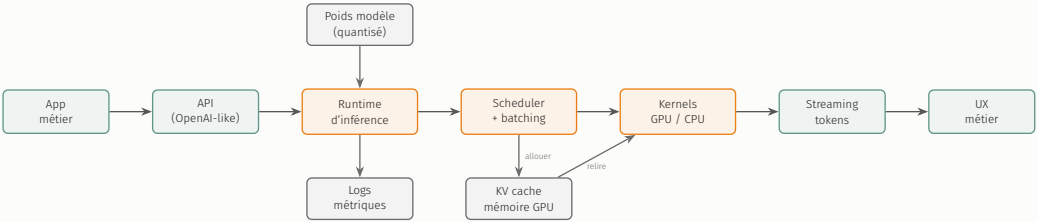
Sources : docs.ollama.com, github.com/ggml-org/llama.cpp, docs.vllm.ai, huggingface.co/docs/text-generation-inference.

Workflow d'une requête SLM en production



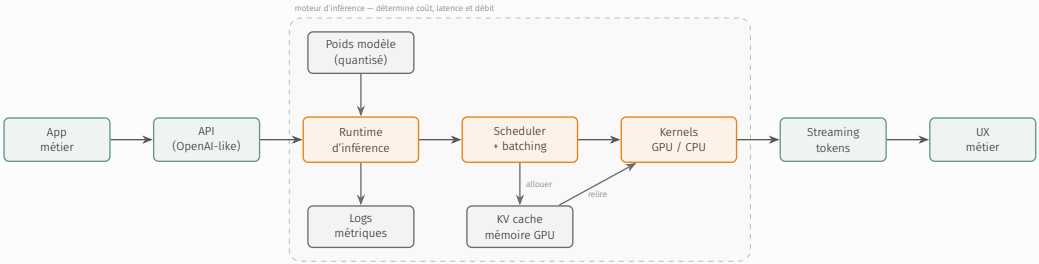
1. Le prompt transite de l'app jusqu'aux kernels GPU/CPU.

Workflow d'une requête SLM en production



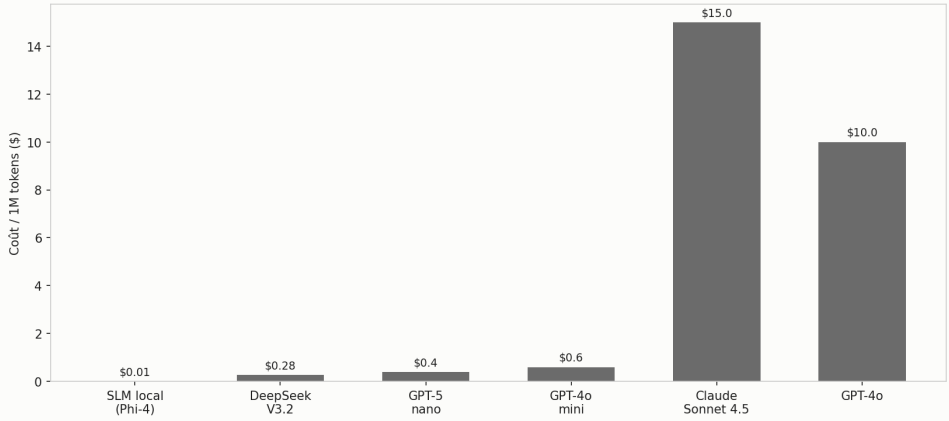
2. Les poids quantisés, le KV cache et les métriques sont gérés par le moteur.

Workflow d'une requête SLM en production



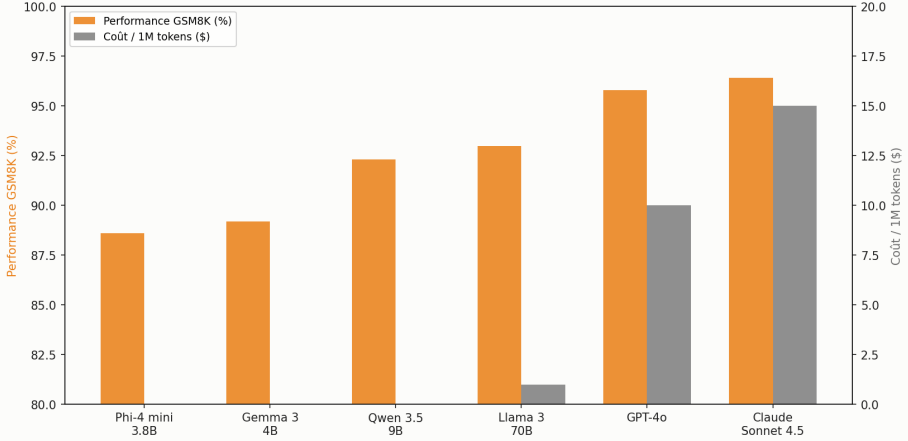
Ce qui compte vraiment : format des poids, quantization, batching, KV cache, streaming, métriques.

Coût local vs coût API



Sources : OpenAI (GPT-4o : $10/M output), Anthropic, DeepSeek · Dell/ESG On-Prem Study 2024

Intelligence par euro, intelligence par watt



Epoch AI (2025) : le prix pour atteindre le niveau GPT-4 a baissé de 40× par an

Conclusion : votre checklist SLM

Ce soir, posez-vous ces questions

1. Mes données peuvent-elles quitter mon infra ?
2. Quelle latence est acceptable pour l'UX ?
3. Quel volume de requêtes par jour/mois ?
4. La tâche est-elle bornée ou ouverte ?

Demain, faites ce test

Prenez un use case → évaluez un LLM (topline) →
testez 2 SLM → mesurez qualité + TTFT + coût.
Le plus petit qui passe le seuil, c'est le bon.

Les 3 erreurs classiques

- **Choisir GPT-4 par défaut** — sans tester plus petit
- **Faire confiance aux benchmarks publics** — construisez le vôtre
- **Fine-tuner trop tôt** — saturez d'abord prompt + RAG

À emporter

- SLM ≠ LLM au rabais — outil différent
- Spécialisation ↑ rapport qualité/coût
- Local = contraintes ≠ cloud

Merci !

Venez échanger sur le stand A2