

# L'intelligence artificielle peut-elle modéliser le langage mathématique?

François CHARTON, Meta AI

# Un échange avec Chat GPT



peut on ranger sept balles rouges et deux balles bleues dans un tube contenant huit balles?



Oui, vous pouvez ranger sept balles rouges et deux balles bleues dans un tube contenant huit balles.  

Il n'y a aucune restriction sur la façon dont vous pouvez disposer les balles dans le tube, du moment que vous y mettez sept balles rouges et deux balles bleues. Vous pouvez par exemple mettre toutes les balles rouges d'un côté et les deux balles bleues de l'autre, ou bien les disposer de manière aléatoire dans le tube.

Il est également possible de mettre sept balles rouges et deux balles bleues dans un tube contenant moins de huit balles, tant que le tube a suffisamment de place pour accueillir toutes les balles. Par exemple, vous pourriez mettre sept balles rouges et deux balles bleues dans un tube contenant sept balles.

# Modèles de langage et mathématiques

- Les modèles de langage sont couramment utilisés pour traiter du langage naturel
  - Traduire, sans dictionnaire, d'une langue vers une autre
  - Dialoguer avec un utilisateur
  - Répondre à des questions précises
- Ils apprennent à partir d'exemples, uniquement
  - Pas besoin de grammaire, de règles linguistiques, de dictionnaire
  - Beaucoup d'exemples (500 milliards de mots pour chat GPT)
- Pourraient-ils faire des mathématiques?

# Modèles de langage et mathématiques

- Des raisons d'espérer:
  - Les mathématiques sont un langage.
  - Il est facile de produire des exemples.
  - Les architectures ont fait leurs preuves: les transformers sont utilisés pour le texte, mais aussi pour l'image et la parole.
- Des raisons de douter:
  - Les réseaux de neurones échouent sur des problèmes très simples comme la multiplication.
  - Nous n'apprenons pas les mathématiques par l'exemple.
  - Peut-on apprendre des mathématiques sans règles?

# Modèles de langage et mathématiques

- Les mathématiques comme problème de traduction
- Entraîner un modèle de langage à traduire un problème (une "phrase" mathématique) en sa solution:
  - $7+9 \Rightarrow 16$
  - $24/16 \Rightarrow 3/2$
  - $\cos(2x) \Rightarrow 2 \cos^2(x) - 1$
  - $x^2-x-1 \Rightarrow \frac{1+\sqrt{5}}{2}, \frac{1-\sqrt{5}}{2}$

# Modèles de langage et mathématiques

## Le programme

- I. Représenter problèmes et solutions comme des “phrases” d’un langage
- II. Générer beaucoup d’exemples: paires de problèmes et de solutions
- III. Entraîner un transformer à prédire les solutions à partir des problèmes

# Les mathématiques comme langage

- Symboles et variables sont des mots :  $\sin$ ,  $\cos$ ,  $x$ ,  $y$ ,  $\pi$ ,  $\gamma$
- Vecteurs et matrices sont des séquences
- Un graphe est une séquence de paires de noeuds
  
- Deux difficultés: représenter les nombres, et les expressions (fonctions, équations)



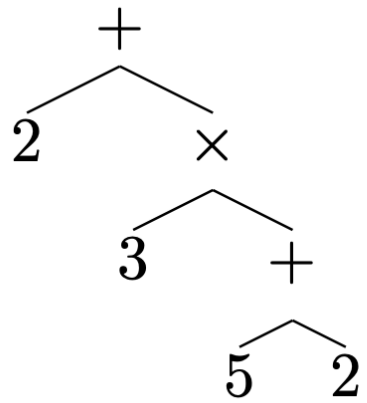
# Représenter les nombres

- Nombres entiers: séquences de chiffres dans une base  $b$ : 1024
  - +, 1, 0, 2, 4 en base 10
  - +, 10, 24 en base 100
  - +, 1, 24 en base 1000
- Rationnels: paires de nombres entiers (deux séquences concaténées)
- Nombres réels: codés en représentation flottante, en base 10:
  - Signe/mantisse/exposant : 3.14 => +, 314, E-2
  - Représentation approchée

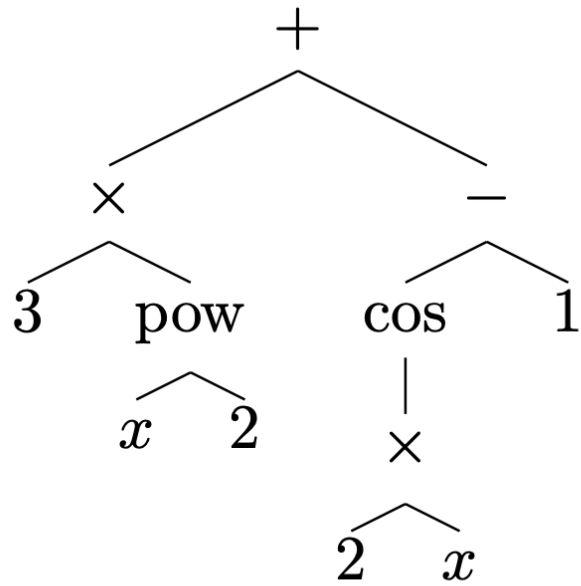


# Représenter des expressions

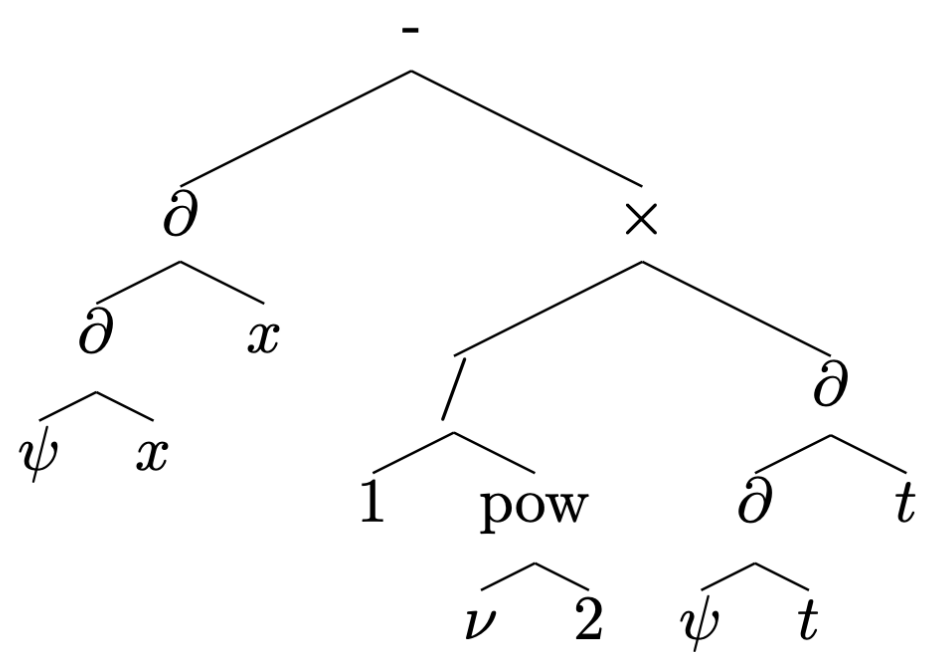
$$2 + 3 \times (5 + 2)$$



$$3x^2 + \cos(2x) - 1$$



$$\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{\nu^2} \frac{\partial^2 \psi}{\partial t^2}$$

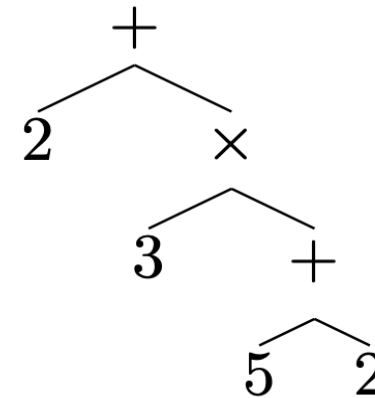


# Les arbres comme séquences

- Notation polonaise

- Commencer à la racine
- Parent avant enfants
- Branche gauche avant branche droite

$$2 + 3 \times (5 + 2)$$



$$+ 2 \times 3 + 5 2$$

# Les expressions comme séquences

Utilisables par les modèles de langage

$$2 + 3 \times (5 + 2)$$

$$+ 2 * 3 + 5 2$$

$$3x^2 + \cos(2x) - 1$$

$$+ * 3 \text{ pow } x 2 - \cos * 2 x 1$$

$$\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{\nu^2} \frac{\partial^2 \psi}{\partial t^2}$$

$$- \partial \partial \psi x x * / 1 \text{ pow } \nu 2 \partial \partial \psi t t$$

# Générer des exemples – deux approches

- Directe
  - Générer un problème P
  - En calculer la solution S
  - Ajouter (P,S) à l'échantillon
  - Nécessite un outil externe
  - Limité aux problèmes que l'on sait résoudre
- Inverse
  - Générer une solution S
  - Trouver un problème P dont S est la solution
  - Ajouter (P,S) à l'échantillon

# Comment générer un problème?

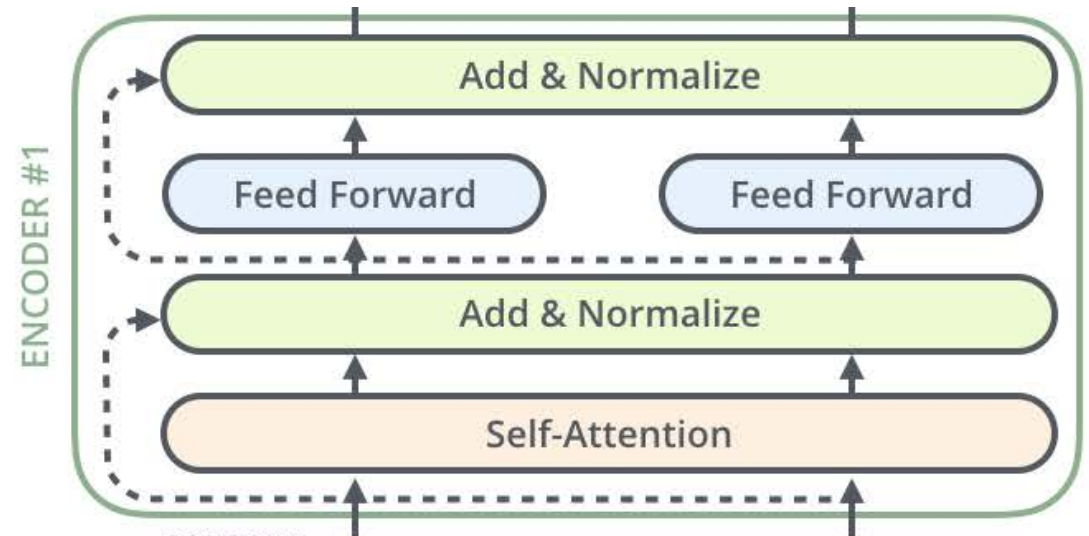
- Un problème = une expression = un arbre
- On utilise la représentation en arbre
  - Générer un arbre au hasard
  - Choisir des opérateurs (dans une liste dépendant du problème) pour les noeuds
  - Choisir des variables, des symboles ou des nombres pour les feuilles
  - Enumérer pour obtenir une expression

# Entraîner des modèles - les transformers

- Modèle destiné au traitement du langage naturel
- Traite des séquences de mots
- Gère les relations entre mots de la phrase, via un mécanisme d'attention

# Les transformers

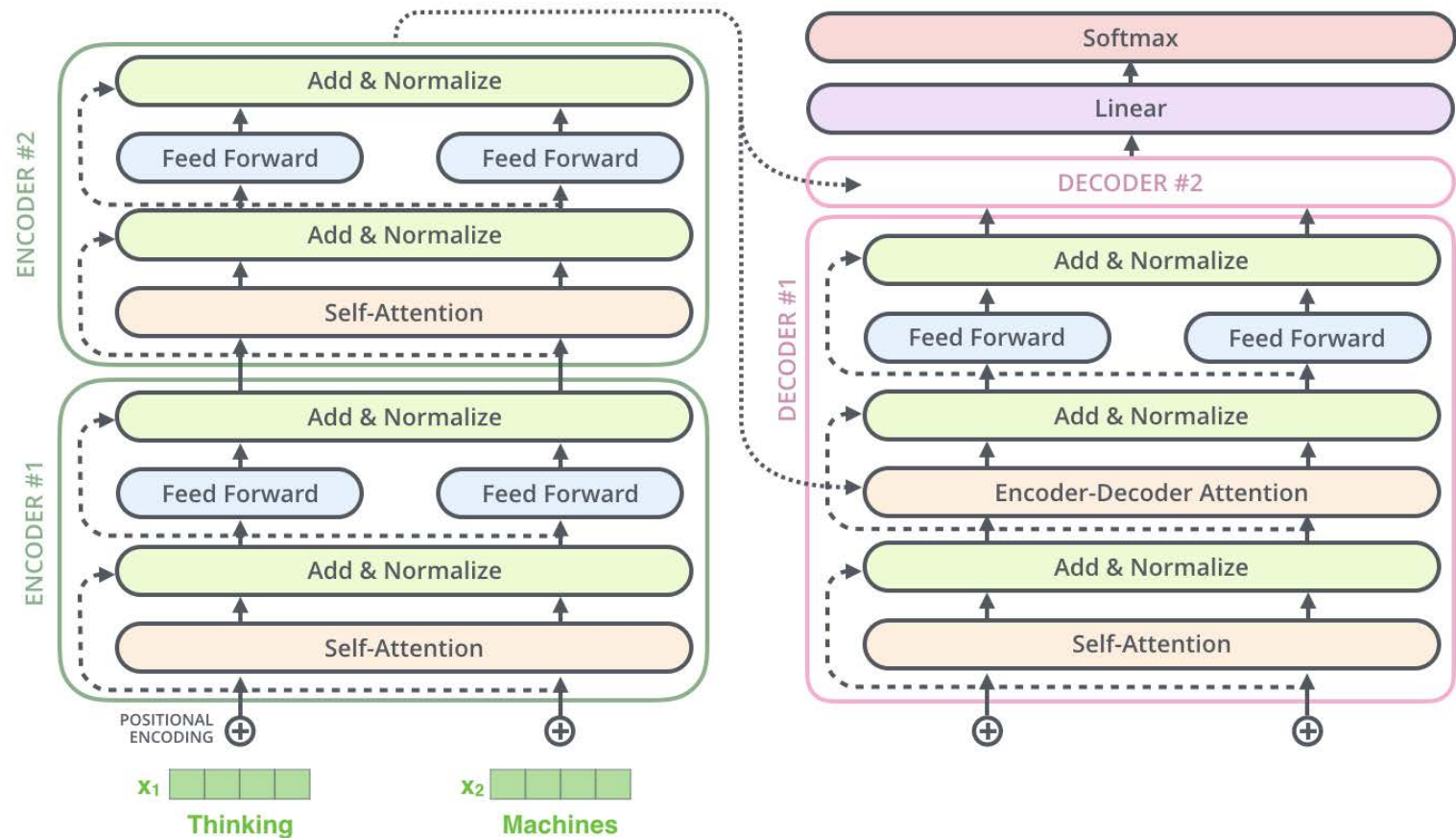
- Une architecture pour le langage
- La couche d'attention prend en compte les auto-corrélations
- Le réseau connecté mémorise et calcule
- La normalisation et les connexions résiduelles assurent la stabilité





# L'architecture de traduction

- Deux transformers côte-à-côte
- L'encodeur traite le langage d'entrée
- Le décodeur produit la traduction mot par mot (auto-régressif)
- Ils sont connectés par un mécanisme d'attention croisée



# Apprentissage

- Le modèle est entraîné sur des données générées
  - Minimise une fonction de coût indépendante du problème (entropie croisée des mots)
  - Le modèle ne connaît pas les mathématiques
- Testé sur des données non vues à l'entraînement
- Les solutions sont vérifiées avec un outil extérieur
  - Sur des critères mathématiques
- Beam search: on génère plusieurs solutions

# Deep learning for symbolic mathematics

(Lample, Charton, 2019)

- Deux problèmes de calcul formel
- Integration: étant donnée  $f$ , trouver une primitive

$$\frac{\cos(2x)}{\sin(x)} \longrightarrow \frac{\log(\cos(x) - 1)}{2} - \frac{\log(\cos(x) + 1)}{2} + 2 \cos(x)$$

- Equations différentielles: résoudre  $G(y, y', x) = 0$  ou  $G(y, y', y'', x) = 0$

$$162x \log(x) y' + 2y^3 \log(x)^2 - 81y \log(x) + 81y = 0 \longrightarrow \frac{9\sqrt{x}}{\sqrt{c + 2x\sqrt{\log(x)}}}$$

# Deep Learning for symbolic mathematics

- Deux problèmes avancés: niveau Terminale/prépa
- Difficiles pour des humains: ne se résolvent pas de tête
- Difficiles pour les machines
  - Algorithme de Risch pour l'intégration
  - Pas d'algorithme pour les équations différentielles
- Implique de la reconnaissance de formes
  - Favorables aux réseaux de neurones

# Génération des données: intégration

Trois méthodes, pour trois échantillons:

- Directe:

- Générer  $f(x)$  au hasard
- Calculer  $F(x)$  avec un logiciel (Sympy)

- Inverse:

- Générer  $F(x)$
- Calculer  $f(x)=F'(x)$

- Par parties:

- Générer  $F(x)$  et  $G(x)$
- Calculer  $f(x)$  et  $g(x)$ , leurs dérivées

- Si  $fG$  est connue, on obtient  $Fg$  par 
$$\int Fg = FG - \int fG$$

# Génération directe

Functions and their primitives generated with the forward approach (FWD)

$\cos^{-1}(x)$	$x \cos^{-1}(x) - \sqrt{1 - x^2}$
$x(2x + \cos(2x))$	$\frac{2x^3}{3} + \frac{x \sin(2x)}{2} + \frac{\cos(2x)}{4}$
$\frac{x(x+4)}{x+2}$	$\frac{x^2}{2} + 2x - 4 \log(x+2)$
$\frac{\cos(2x)}{\sin(x)}$	$\frac{\log(\cos(x) - 1)}{2} - \frac{\log(\cos(x) + 1)}{2} + 2 \cos(x)$
$3x^2 \sinh^{-1}(2x)$	$x^3 \sinh^{-1}(2x) - \frac{x^2 \sqrt{4x^2 + 1}}{6} + \frac{\sqrt{4x^2 + 1}}{12}$
$x^3 \log(x^2)^4$	$\frac{x^4 \log(x^2)^4}{4} - \frac{x^4 \log(x^2)^3}{2} + \frac{3x^4 \log(x^2)^2}{4} - \frac{3x^4 \log(x^2)}{4} + \frac{3x^4}{8}$

# Génération inverse

---

Functions and their primitives generated with the backward approach (BWD)

---

$$\cos(x) + \tan^2(x) + 2$$

$$\frac{1}{x^2 \sqrt{x-1} \sqrt{x+1}}$$

$$\left( \frac{2x}{\cos^2(x)} + \tan(x) \right) \tan(x)$$

$$\frac{x \tan\left(\frac{e^x}{x}\right) + \frac{(x-1)e^x}{\cos^2\left(\frac{e^x}{x}\right)}}{x}$$

$$1 + \frac{1}{\log(\log(x))} - \frac{1}{\log(x) \log(\log(x))^2}$$

$$-2x^2 \sin(x^2) \tan(x) + x(\tan^2(x) + 1) \cos(x^2) + \cos(x^2) \tan(x)$$

$$x + \sin(x) + \tan(x)$$

$$\frac{\sqrt{x-1} \sqrt{x+1}}{x}$$

$$x \tan^2(x)$$

$$x \tan\left(\frac{e^x}{x}\right)$$

$$x + \frac{x}{\log(\log(x))}$$

$$x \cos(x^2) \tan(x)$$



# Génération par parties

---

Functions and their primitives generated with the integration by parts approach (IBP)

---

$$x(x + \log(x))$$

$$\frac{x}{(x+3)^2}$$

$$\frac{x + \sqrt{2}}{\cos^2(x)}$$

$$x(2x + 5)(3x + 2 \log(x) + 1)$$

$$\frac{\left(x - \frac{2x}{\sin^2(x)} + \frac{1}{\tan(x)}\right) \log(x)}{\sin(x)}$$

$$x^3 \sinh(x)$$

$$\frac{x^2(4x + 6 \log(x) - 3)}{12}$$

$$\frac{-x + (x+3) \log(x+3)}{x+3}$$

$$(x + \sqrt{2}) \tan(x) + \log(\cos(x))$$

$$\frac{x^2(27x^2 + 24x \log(x) + 94x + 90 \log(x))}{18}$$

$$\frac{x \log(x) + \tan(x)}{\sin(x) \tan(x)}$$

$$x^3 \cosh(x) - 3x^2 \sinh(x) + 6x \cosh(x) - 6 \sinh(x)$$

# Génération

- 5 échantillons, 3 pour l'intégration, 2 pour les équations différentielles (d'ordre 1 et 2)
- Opérateurs: les quatre opérations, et les fonctions élémentaires (exp, log, sin cos, tan, arcsin, arccos, arctan, sqrt, pow)
- Une variable, constantes entières entre -5 et 5

	Forward	Backward	Integration by parts	ODE 1	ODE 2
Training set size	20M	40M	20M	40M	40M
Input length	18.9±6.9	70.2±47.8	17.5±9.1	123.6±115.7	149.1±130.2
Output length	49.6±48.3	21.3±8.3	26.4±11.3	23.0±15.2	24.3±14.9
Length ratio	2.7	0.4	2.0	0.4	0.1
Input max length	69	450	226	508	508
Output max length	508	75	206	474	335

# Résultats

	Integration (FWD)	Integration (BWD)	Integration (IBP)	ODE (order 1)	ODE (order 2)
Beam size 1	93.6	98.4	96.8	77.6	43.0
Beam size 10	95.6	99.4	99.2	90.5	73.0
Beam size 50	96.2	99.7	99.5	94.0	81.2

- Intégration
  - Près de 100%, même sans beam search
  - Indépendamment de la méthode de génération
- Equations différentielles
  - Problèmes plus difficiles, performance plus faible
  - Utilité de la beam search

# Comparaisons avec Mathematica

	Integration (BWD)	ODE (order 1)	ODE (order 2)
Mathematica (30s)	84.0	77.2	61.6
Matlab	65.2	-	-
Maple	67.4	-	-
Beam size 1	98.4	81.2	40.8
Beam size 10	99.6	94.0	73.2
Beam size 50	99.6	97.0	81.0

# Limites du modèle

- Le modèle est entraîné sur des données générées
- Les données d'entraînement et de test proviennent du même générateur
- Que se passe-t-il si on teste le modèle sur des données différentes?

# Limites du modèle

Training data	Forward (FWD)			Backward (BWD)		
	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50
FWD	93.6	95.6	96.2	10.9	13.9	17.2
BWD	18.9	24.6	27.5	98.4	99.4	99.7

- Le modèle est très sensible à la distribution des données
- Ne généralise plus quand les données de test proviennent d'une distribution différente de celle d'entraînement

# Génération directe

Functions and their primitives generated with the forward approach (FWD)

$\cos^{-1}(x)$	$x \cos^{-1}(x) - \sqrt{1 - x^2}$
$x(2x + \cos(2x))$	$\frac{2x^3}{3} + \frac{x \sin(2x)}{2} + \frac{\cos(2x)}{4}$
$\frac{x(x+4)}{x+2}$	$\frac{x^2}{2} + 2x - 4 \log(x+2)$
$\frac{\cos(2x)}{\sin(x)}$	$\frac{\log(\cos(x) - 1)}{2} - \frac{\log(\cos(x) + 1)}{2} + 2 \cos(x)$
$3x^2 \sinh^{-1}(2x)$	$x^3 \sinh^{-1}(2x) - \frac{x^2 \sqrt{4x^2 + 1}}{6} + \frac{\sqrt{4x^2 + 1}}{12}$
$x^3 \log(x^2)^4$	$\frac{x^4 \log(x^2)^4}{4} - \frac{x^4 \log(x^2)^3}{2} + \frac{3x^4 \log(x^2)^2}{4} - \frac{3x^4 \log(x^2)}{4} + \frac{3x^4}{8}$



# Génération inverse

---

Functions and their primitives generated with the backward approach (BWD)

---

$$\cos(x) + \tan^2(x) + 2$$

$$\frac{1}{x^2 \sqrt{x-1} \sqrt{x+1}}$$

$$\left( \frac{2x}{\cos^2(x)} + \tan(x) \right) \tan(x)$$

$$\frac{x \tan\left(\frac{e^x}{x}\right) + \frac{(x-1)e^x}{\cos^2\left(\frac{e^x}{x}\right)}}{x}$$

$$1 + \frac{1}{\log(\log(x))} - \frac{1}{\log(x) \log(\log(x))^2}$$

$$-2x^2 \sin(x^2) \tan(x) + x(\tan^2(x) + 1) \cos(x^2) + \cos(x^2) \tan(x)$$

$$x + \sin(x) + \tan(x)$$

$$\frac{\sqrt{x-1} \sqrt{x+1}}{x}$$

$$x \tan^2(x)$$

$$x \tan\left(\frac{e^x}{x}\right)$$

$$x + \frac{x}{\log(\log(x))}$$

$$x \cos(x^2) \tan(x)$$

# Limites du modèle

Training data	Forward (FWD)			Backward (BWD)			Integration by parts (IBP)		
	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50
FWD	93.6	95.6	96.2	10.9	13.9	17.2	85.6	86.8	88.9
BWD	18.9	24.6	27.5	98.4	99.4	99.7	42.9	54.6	59.2
BWD + IBP	41.6	54.9	56.1	98.2	99.4	99.7	96.8	99.2	99.5

- Sur l'échantillon "par parties", les modèles directs et inverses ont de bonnes performances
- La généralisation est possible tant qu'on ne s'éloigne pas trop de la distribution d'entraînement
- Diversifier les générateurs améliore les performances

# Learning advanced computations from examples

(Charton, Hayat, Lample, 2020)

- Systèmes différentiels:  $n$  équations différentielles du premier ordre, en  $n$  variables:  $\frac{dx}{dt} = f(x) \quad x \in R^n$
- La plupart n'ont pas de solution analytique, mais les propriétés des trajectoires au voisinage des points critiques ( $f(x)=0$ ) sont données par la théorie
- Théorème spectral: soit  $J(x) = \nabla f(x)$ , le Jacobien de  $f$ . Si toutes ses valeurs propres (complexes) en un point critique ont des parties réelles négatives, l'équilibre est stable, et la plus grande partie réelle est la vitesse de convergence.

# Stabilité locale d'un système différentiel

- Système initial

$$\begin{aligned}\frac{dx_1}{dt} &= \cos(x_2) - 1 - \sin(x_1) \\ \frac{dx_2}{dt} &= x_1^2 - \sqrt{1 + x_2}\end{aligned}$$

- Différencier

$$J(x) = \begin{pmatrix} -\cos(x_1) & -\sin(x_2) \\ 2x_1 & -(2\sqrt{1+x_2})^{-1} \end{pmatrix}$$

- Evaluer

$$x_e = (0.1, \dots, 0.1) \in \mathbb{R}^n, \quad J(x_e) = \begin{pmatrix} -\cos(0.1) & -\sin(0.1) \\ 0.2 & -(2\sqrt{1+0.1})^{-1} \end{pmatrix},$$

- Calcul des valeurs propres

$$\lambda_1 = -1.031, \quad \lambda_2 = -0.441$$

- Prendre la plus grande: -0.441, système stable localement

# Stabilité locale d'un système différentiel

Table 1: Prediction of local convergence speed (within 10%).

	Degree 2	Degree 3	Degree 4	Degree 5	Degree 6	Overall
4 layers, dim 512	88.0	74.3	63.8	54.2	45.0	65.1
6 layers, dim 512	93.6	85.5	77.4	71.5	64.9	78.6
8 layers, dim 512	95.3	88.4	83.4	79.2	72.4	83.8
4 layers, dim 1024	91.2	80.1	71.6	61.8	54.4	71.9
6 layers, dim 1024	95.7	89.0	83.4	78.4	72.6	83.8
8 layers, dim 1024	<b>96.3</b>	<b>90.4</b>	<b>86.2</b>	<b>82.7</b>	<b>77.3</b>	<b>86.6</b>

# Contrôlabilité d'un système différentiel

- Si le système est sur-paramétré, via des variables de contrôle  $u$

$$\frac{dx}{dt} = f(x, u) \quad x \in R^n \quad u \in R^p$$

- Peut on choisir  $u$  tel que :  $x(0)=x_0$  et  $x(T)=x_T$
- Condition de Kalman: si  $A = \partial_x f(x, u)$   $B = \partial_u f(x, u)$ , le système est contrôlable si  $\text{rang}[A, AB, \dots, A^{n-1}B]=n$

# Contrôlabilité d'un système différentiel

- Partant d'un système

$$\frac{dx_1(t)}{dt} = \sin(x_1^2) + \log(1 + x_2) + \frac{\text{atan}(ux_1)}{1 + x_2}$$

$$\frac{dx_2(t)}{dt} = x_2 - e^{x_1x_2},$$

- Différencier selon x

$$A(x, u) = \begin{pmatrix} 2x_1 \cos(x_1^2) + \frac{u(1+x_2)^{-1}}{1+u^2x_1^2} & (1+x_2)^{-1} - \frac{\text{atan}(ux_1)}{(1+x_2)^2} \\ -x_2 e^{x_1x_2} & 1 - x_1 e^{x_1x_2} \end{pmatrix}$$

- Différencier selon u

$$B(x, u) = \begin{pmatrix} x_1((1 + u^2x_1^2)(1 + x_2))^{-1} \\ 0 \end{pmatrix}$$

- Evaluer

$$A(x_e, u_e) = \begin{pmatrix} 1.50 & 0.46 \\ -0.64 & 0.36 \end{pmatrix}, \quad B(x_e, u_e) = \begin{pmatrix} 0.27 \\ 0 \end{pmatrix}$$



# Contrôlabilité d'un système différentiel

- Evaluer  $A(x_e, u_e) = \begin{pmatrix} 1.50 & 0.46 \\ -0.64 & 0.36 \end{pmatrix}$ ,  $B(x_e, u_e) = \begin{pmatrix} 0.27 \\ 0 \end{pmatrix}$
- Calculer la matrice de contrôle  $C = [B, AB]((x_e, u_e)) = \begin{pmatrix} 0.27 & 0.40 \\ 0 & -0.17 \end{pmatrix}$
- Calculer son rang  $\text{rank}(C) = 2$

# Contrôlabilité d'un système différentiel

Table 3: Accuracy of autonomous control task over a balanced sample of systems with 3 to 6 equations.

	Dimension 64	Dimension 128	Dimension 256	Dimension 512
1 layers	81.0	85.5	88.3	90.4
2 layers	82.7	88.0	93.9	95.5
4 layers	84.1	89.2	95.6	96.9
6 layers	84.2	90.7	96.3	<b>97.4</b>

- Un modèle assez large (256 dimensions, 2 couches) prévoit la contrôlabilité dans 95% des cas
- De très petits modèles, insuffisants pour le langage naturel, ont des performances élevées

# Calcul de la matrice de contrôle

- Pour un système contrôlable, trouver  $K$ , tel que  $u=Kx$
- Une solution est

$$K = -B^{tr} \left( e^{-AT} \left[ \int_0^T e^{-At} B B^{tr} e^{-A^{tr}t} dt \right] e^{-A^{tr}T} \right)^{-1}$$

- Ici  $K = (-22.8, 44.0)$

# Calcul de la matrice de contrôle

**Table 5: Prediction of feedback matrices - Approximation vs. correct mathematical feedback.**

	Degree 3	Degree 4	Degree 5	Degree 6	Overall
Prediction within 10%	50.0	9.3	2.1	0.4	15.8
Correct feedback matrix	87.5	77.4	58.0	41.5	66.5

- La matrice  $K$  peut être prédite pour les petits degrés, mais la performance est moyennes
- En revanche, le modèle prévoit une solution correcte dans 2 cas sur 3
- Les mathématiques sous-jacentes ont été apprises

# Deep symbolic regression for recurrent sequences

(d'Ascoli, Kamienny, Lample, Charton 2022)

- Test de QI: 1, 2, 4, 7, 11, 16, quel est le prochain terme?
- Deux approches:
  - Régression numérique: prédiction directe du terme suivant
  - Régression symbolique: trouver une formule, puis l'utiliser pour prédire
    - Une formule fermée :  $u_n = n(n+1) / 2 + 1$
    - Ou une récurrence:  $u_n = u_{n-1} + n$
- On considère deux cas: suites entières, ou réelles

# Générer des récurrences

- On génère une fonction  $f$ , des  $n$  et/ou des  $u_{n-k}$
- On l'initialise sur quelques valeurs au hasard
- On utilise  $f$  pour calculer les valeurs suivantes
  
- Dans le cas numérique, on veut prévoir  $(u_p, \dots, u_{p+q-1})$  à partir de  $(u_0, \dots, u_{p-1})$
- Dans le cas symbolique, on prévoit  $f$  à partir de  $(u_0, \dots, u_{p-1})$

# Evaluer le modèle

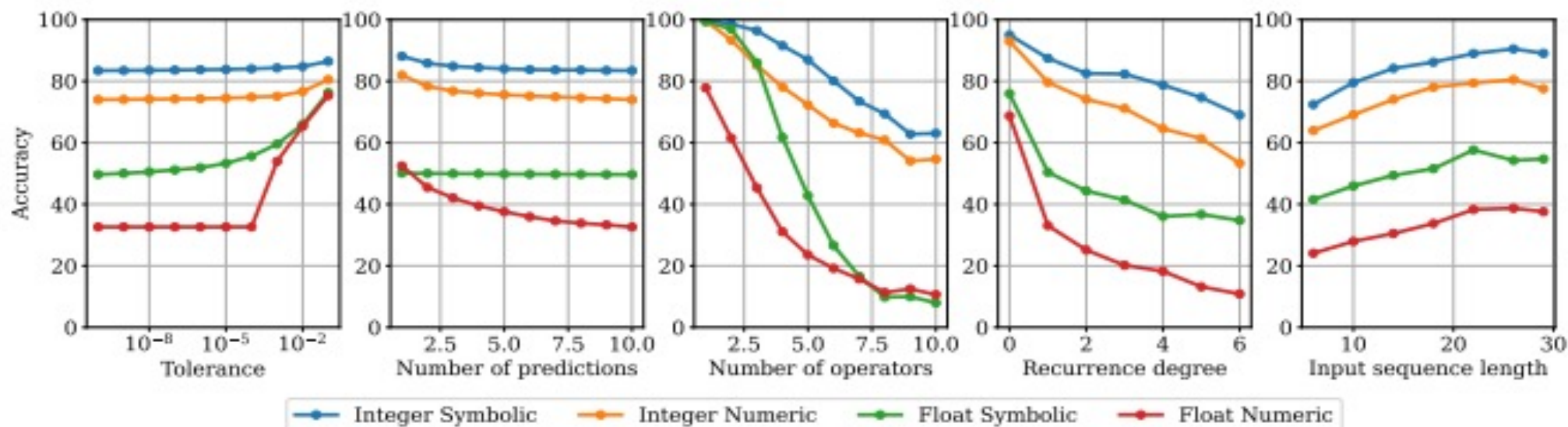
- La performance est mesurée sur les 10 prochains termes de la suite
- Évaluée sur un jeu de test de 10 000 exemples

$$\text{acc}(n_{\text{pred}}, \tau) = \mathbb{P} \left( \max_{1 \leq i \leq n_{\text{pred}}} \left| \frac{\hat{u}_i - u_i}{u_i} \right| < \tau \right)$$

# Résultats

Model	Integer		Float	
	$n_{op} \leq 5$	$n_{op} \leq 10$	$n_{op} \leq 5$	$n_{op} \leq 10$
Symbolic	<b>92.7</b>	<b>78.4</b>	<b>74.2</b>	<b>43.3</b>
Numeric	83.6	70.3	45.6	29.0

Table 6: Average in-distribution accuracies of our models. We set  $\tau = 10^{-10}$  and  $n_{pred} = 10$ .





# Résultats hors distribution (OEIS)

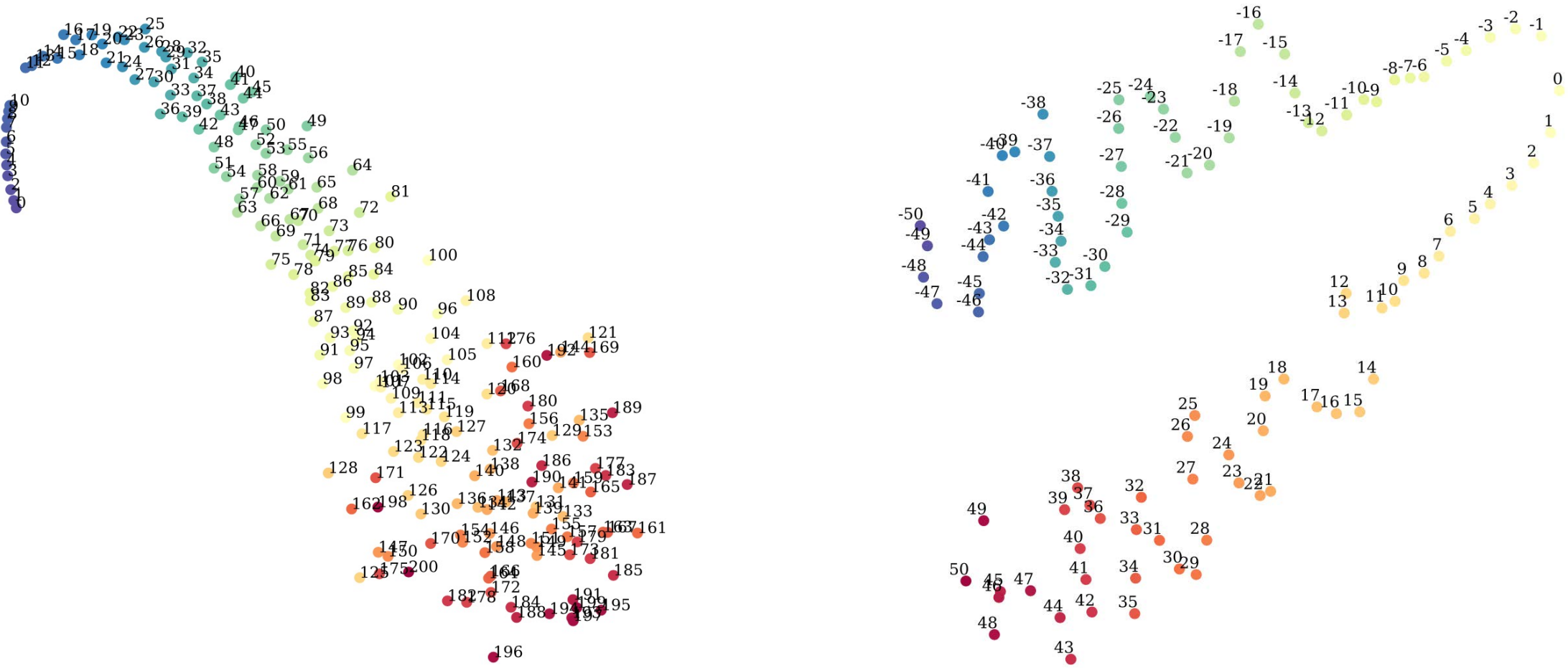
Model	$n_{input} = 15$		$n_{input} = 25$	
	$n_{pred} = 1$	$n_{pred} = 10$	$n_{pred} = 1$	$n_{pred} = 10$
Symbolic (ours)	33.4	19.2	34.5	21.3
Numeric (ours)	53.1	27.4	54.9	29.5
FindSequenceFunction	17.1	12.0	8.1	7.2
FindLinearRecurrence	17.4	14.8	21.2	19.5

Table 7: **Accuracy of our integer models and Mathematica functions on OEIS sequences.** We use as input the first  $n_{input} = \{15, 25\}$  first terms of OEIS sequences and ask each model to predict the next  $n_{pred} = \{1, 10\}$  terms. We set the tolerance  $\tau = 10^{-10}$ .

# Résultats hors distribution (OEIS)

OEIS	Description	First terms	Predicted recurrence
A000792	$a(n) = \max\{(n - i)a(i), i < n\}$	1, 1, 2, 3, 4, 6, 9, 12, 18, 27	$u_n = u_{n-1} + u_{n-3} - u_{n-1} \% u_{n-3}$
A000855	Final two digits of $2^n$	1, 2, 4, 8, 16, 32, 64, 28, 56, 12	$u_n = (2u_{n-1}) \% 100$
A006257	Josephus sequence	0, 1, 1, 3, 1, 3, 5, 7, 1, 3	$u_n = (u_{n-1} + n) \% (n - 1) - 1$
A008954	Final digit of triangular number $n(n + 1)/2$	0, 1, 3, 6, 0, 5, 1, 8, 6, 5	$u_n = (u_{n-1} + n) \% 10$
A026741	$a(n) = n$ if $n$ odd, $n/2$ if $n$ even	0, 1, 1, 3, 2, 5, 3, 7, 4, 9	$u_n = u_{n-2} + n // (u_{n-1} + 1)$
A035327	$n$ in binary, switch 0's and 1's, back to decimal	1, 0, 1, 0, 3, 2, 1, 0, 7, 6	$u_n = (u_{n-1} - n) \% (n - 1)$
A062050	$n$ -th chunk consists of the numbers $1, \dots, 2^n$	1, 1, 2, 1, 2, 3, 4, 1, 2, 3	$u_n = (n \% (n - u_{n-1})) + 1$
A074062	Reflected Pentanacci numbers	5, -1, -1, -1, -1, 9, -7, -1, -1, -1	$u_n = 2u_{n-5} - u_{n-6}$

# Représentation interne des nombres



# Le modèle apprend -il des mathématiques?

Constant	Approximation	Rel. error
0.3333	$(3 + \exp(-6))^{-1}$	$10^{-5}$
0.33333	$1/3$	$10^{-5}$
3.1415	$2 \arctan(\exp(10))$	$10^{-7}$
3.14159	$\pi$	$10^{-7}$
1.6449	$1 / \arctan(\exp(4))$	$10^{-7}$
1.64493	$\pi^2 / 6$	$10^{-7}$
0.123456789	$10/9^2$	$10^{-9}$
0.987654321	$1 - (1/9)^2$	$10^{-11}$

Approcher des constantes réelles

Expression $u_n$	Approximation $\hat{u}_n$
$\operatorname{arcsinh}(n)$	$\log(n + \sqrt{n^2 + 1})$
$\operatorname{arccosh}(n)$	$\log(n + \sqrt{n^2 - 1})$
$\operatorname{arctanh}(1/n)$	$\frac{1}{2} \log(1 + 2/n)$
$\operatorname{catalan}(n)$	$u_{n-1}(4 - 6/n)$
$\operatorname{dawson}(n)$	$\frac{n}{2n^2 - u_{n-1} - 1}$
$j_0(n)$ (Bessel)	$\frac{\sin(n) + \cos(n)}{\sqrt{\pi n}}$
$i_0(n)$ (mod. Bessel)	$\frac{e^n}{\sqrt{2\pi n}}$

Approcher des fonctions inconnues

# Deux questions pour les modèles de langage

- Les mathématiques sont-elles apprises?
- Le modèle raisonne-t-il comme nous?

# Les mathématiques sont-elles apprises?

- Si on donne au modèle des séries de 4 entiers,  $a, b, c, d$ 
  - Il apprendra à prévoir si  $a/b < c/d$
  - Il n'apprendra pas à calculer  $a/b+c/d$ , ou  $ac/bd$
- Avec deux entiers positifs  $a$  et  $b$ 
  - Il n'apprendra pas à simplifier la fraction  $a/b$

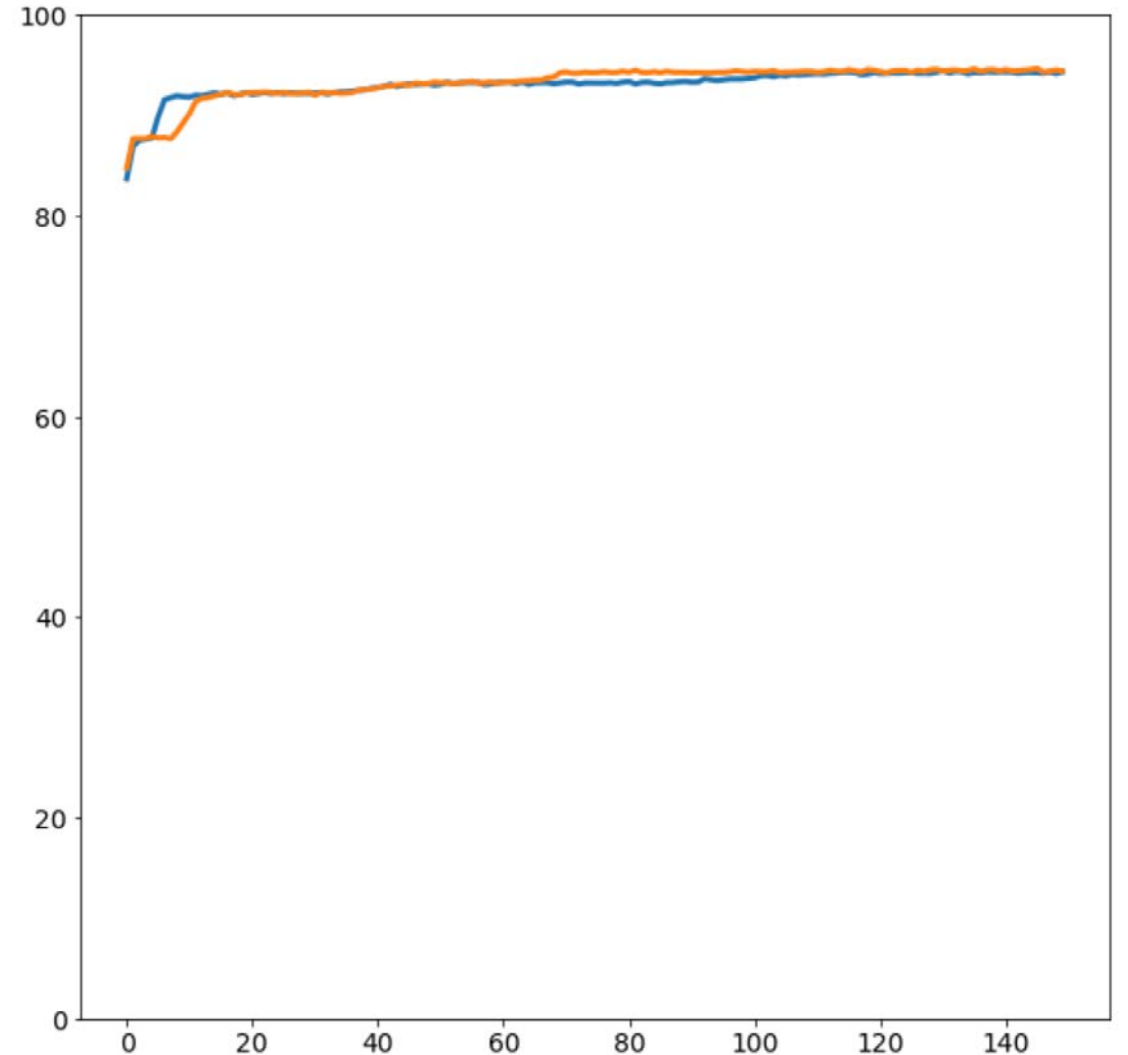
# Apprendre les PGCD

- On génère des paires d'entiers  $(a,b)$  entre 1 et 1 million (16 et 24)
- On calcule leur PGCD (8)
- On les code comme des sequences de chiffres dans une base  $b$
- On entraîne un modèle à “traduire”:
  - $+, 1, 6, +, 2, 4 \Rightarrow +, 8$



# Apprendre les PGCD

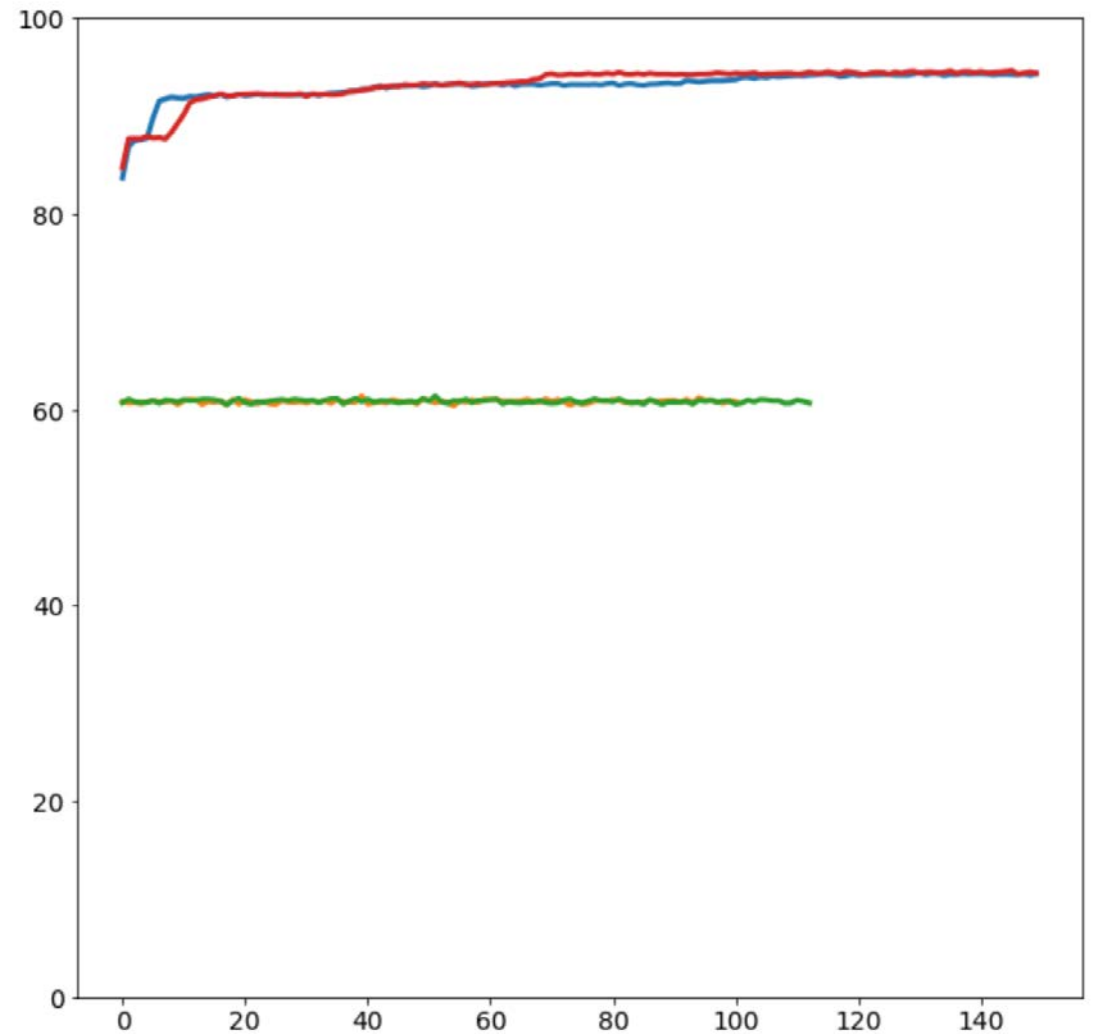
- Base 30
- Transformers 1 couche, 64 dimensions
- 85% après 300 000 exemples
- 95% après 45 millions d'exemples





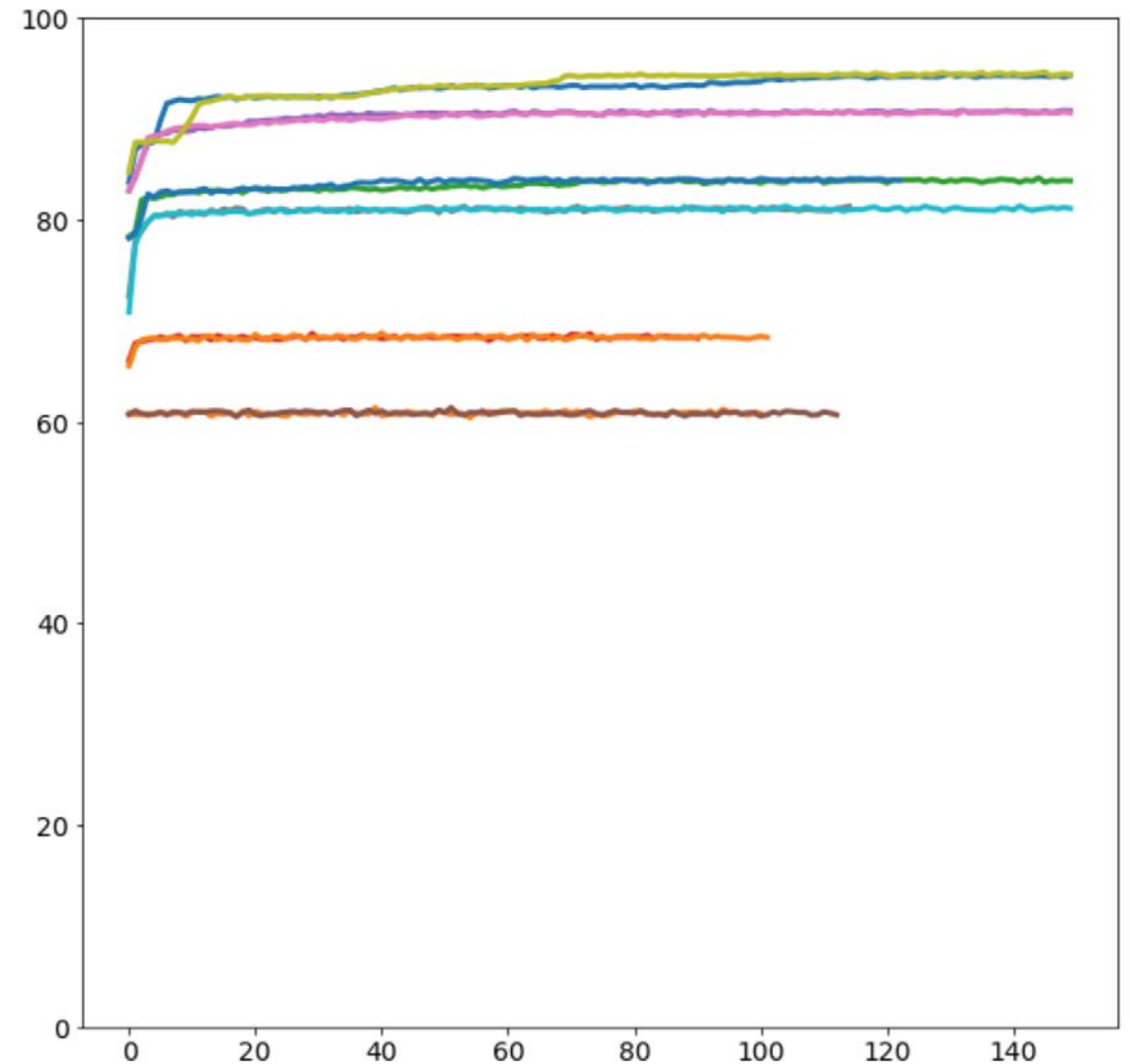
# Apprendre les PGCD

- En base 31
- 61% de prévisions correctes
- Pas d'apprentissage



# Apprendre les PGCD?

- De haut en bas: base 30, 6, 10, 2, 3, 31
- Le PGCD ne devrait pas dépendre de la base
- Les maths sont elles apprises?



# Le PGCD n'est pas appris

- La probabilité qu'un PGCD soit  $k$  est proportionnelle à  $1/k^2$  (Césaro)
- 60.8% des exemples ont des PGCD de 1: valeur par défaut du modèle
- En base 2 et 4, les PGCD 2,4,8,16 sont correctement prévus
  - Pour tous les autres nombres, le modèle prévoit la plus grande puissance de 2 qui divise le PGCD: pour 6 et 12, il prévoit 2
  - Le modèle compte les zéros à droite des opérandes:
    - Pour 11100 (28) et 1110 (14), le modèle prévoit 2 (!!!)
    - 111100 (60) et 111000 (56), le modèle prévoit 4

# Le PGCD n'est pas appris

- En base 30, le modèle prévoit correctement les PGCD égaux à 1,2,3,4,5,6,8,9,10,12...
- En base 31, 1 et 31
- Le modèle exploite les représentations pour calculer les cas faciles, et échoue sur les autres
- Les mathématiques sous-jacentes ne sont pas apprises.

# Linear algebra with transformers

## What is my math transformer doing? (Charton 2022)

- 9 problèmes d'algèbre linéaire
  - Opérations de base sur les matrices (addition, produit scalaire, multiplication, transposition): performance 100%, sur des matrices 5x5 à 30x30
  - Calcul des valeurs propres: 100% sur des matrices 5x5 à 8x8
  - Calcul des vecteurs propres: 95% sur des matrices 5x5
  - Inversion: 90% sur des matrices 5x5
- Les matrices sont codées comme des séquences de coefficients
- Coefficients arrondis à trois chiffres significatifs, codés sur 3 symboles

# Apprendre à diagonaliser

- Etant donnée une matrice symétrique  $M$
- Le modèle prédit un vecteur  $D$ , et une matrice  $H$  tels que
  - $HMH^T = HMH^{-1} = \text{Diag}(D)$
- A partir d'exemples seulement: triplets  $(M, D, H)$
  
- Nous savons grâce à la théorie
  - Que  $D$  est composé des valeurs propres de  $M$  (uniques à une permutation près)
  - Que  $H$  est unitaire, et que ses lignes et ses colonnes
    - sont de norme un
    - Sont deux à deux orthogonales

# Apprendre à diagonaliser

- Un modèle entraîné est testé sur 100 000 matrices, 92% de réponses correctes
  - 8 000 prédictions fausses
- Dans tous les cas sauf 6, les valeurs propres sont prévues avec moins de 1% d'erreur.
- Dans 98.9% des exemples de test, toutes les lignes et les colonnes de  $H$  sont de norme un
- Ces deux propriétés sont vérifiées même quand le modèle se trompe
- Des mathématiques ont été apprises

# Apprendre à diagonaliser

- Ces propriétés sont vérifiées en début d'apprentissage: sur un modèle à moitié entraîné (performance 70%)
  - Les valeurs propres sont correctes dans 99.6% des cas de test
  - Lignes et colonnes de H ont une norme de un dans 96.7%
- Sur des matrices 6x6, les modèles ne prévoient correctement que 46% des exemples de test, mais
  - Les valeurs propres sont correctes dans 99.6% des cas
  - Les lignes et colonnes ont des normes de un dans 93.1%



# Comprendre les erreurs

- Pour la diagonalisation, les erreurs du modèle sont dues à la non-orthogonalité des vecteurs propres (lignes et colonnes de  $H$ )
- Le calcul des valeurs propres, et la norme des vecteurs propres sont apprises dans tout les cas.
- Le modèle, quand il se trompe, ne produit pas de solutions absurdes (hallucinations)
- On peut prévoir les erreurs en calculant le coefficient de conditionnement de  $H$  (rapport des valeurs singulières extrêmes), qui devrait être 1 si  $H$  est orthogonal.
  - $c(H) > 1.045$  prévoit le résultat du modèle dans 99.3% des cas (99.9% des succès et 96.7% des échecs)

# Comprendre les erreurs – inversion de matrices

- Etant donné une matrice  $M$ , on cherche à prévoir  $P$  telle que  $MP \approx Id$ 
  - Les matrices  $M$  sont inversibles, il existe une inverse  $M^{-1}$
- Sur cette tâche, le modèle peine à prévoir plus de 90% des exemples de test.
  - le modèle prévoit  $P \approx M^{-1}$ , mais on n'a pas  $MP \approx Id$
  - le problème est mal conditionné
    - difficulté classique de cette tâche: les algorithmes numériques échoueront aussi
- Le conditionnement de  $M$  ( $c(M) > 66$ ) prévoit 98% des prévisions du modèle (uniquement en fonction de l'entrée)
- Le modèle échoue pour de bonnes raisons mathématiques
- Ses échecs sont prévisibles

# Performances hors-distribution – calcul de valeurs propres

- Un modèle prévoit 100% des valeurs propres de matrices symétriques 5x5 à 8x8
- Matrices symétriques à coefficients indépendants
- Matrices de Wigner, les valeurs propres sont distribuées selon une loi du demi-cercle
  - Symétrique autour de l'origine
  - Variance dépendant de la variance des coefficients et de la dimension
  - Support borne
- Peut on généraliser à des matrices non-Wigner?

# Calcul de valeurs propres – d'autres générateurs

- Pour modifier la distribution des valeurs propres
  - On génère des matrices de Wigner,  $M$ , à coefficients gaussiens
    - Garantit que les orientations des vecteurs propres soient uniformément distribuées
  - On les diagonalise  $M=HDH^T$ 
    - Les composantes de  $D$  suivent un demi-cercle
  - On remplace  $D$  par  $E$ , ayant la distribution de notre choix
  - On entraîne (ou teste) le modèle sur  $M'=HEH^T$
- On considère 6 distributions de valeurs propres
  - Symétriques: Laplace, Normale, Uniforme
  - Positives: valeurs absolues de demi-cercle et Laplace
  - Marchenko-Pastur: distribution des valeurs propres de  $M^T M$

# Calculs de valeurs propres

	Semi-circle	Uniform	Gaussian	Laplace	abs-sc	abs-Lapl	Marchenko
Semi-circle	100	34	36	39	1	5	0
Uniform	93	100	76	70	92	70	2
Gaussian	100	100	100	100	100	100	99
Laplace	100	100	100	100	100	100	100
Abs-semicircle	0	5	4	4	100	78	20
Abs-Laplace	0	4	5	5	100	100	100
Marchenko-Pastur	0	4	4	4	100	76	100

Table 1: **Out-of-distribution generalization. Eigenvalues of 5x5 matrices.** Rows are the training distributions, columns the test distributions.

- Gauss et Laplace généralisent à Wigner (pas l'inverse)
- Généralisation loin de la distribution d'entraînement
  - Aux matrices définies positives

# Calcul de valeurs propres

- Les distributions robustes apprennent plus vite

	Semi-circle	Uniform	Gaussian	Laplace	abs-sc	abs-Lapl	Marchenko
8x8 matrices							
Semicircle	0	0	0	0	0	0	0
Uniform	91	100	65	57	89	55	0
Gaussian	100	100	100	99	100	99	41
Laplace	100	100	100	100	100	100	97
Abs-semicircle	0	1	1	0	100	53	0
Abs-Laplace	0	1	1	1	100	100	98
Marchenko-Pastur	0	0	0	0	1	1	20
10x10 matrices							
Gaussian (12/1 layers)	100	100	100	98	100	97	3
Laplace (8/1 layers)	100	100	100	100	100	100	74

**Table 2: Out-of-distribution generalization. Eigenvalues of 8x8 and 10x10 matrices, accuracy after 36 million examples.** Rows are the training distributions, columns the test distributions.

# Conclusions

- Les machines peuvent apprendre les mathématiques
  - Un nouveau domaine de recherche
  - Applications aux sciences
- Les mathématiques peuvent aider à comprendre l'apprentissage automatique
- Il y a des points communs entre apprentissage humain et mécanique
  - Algèbre linéaire: mêmes mathématiques
- Mais des différences profondes sur ce qui est facile et difficile





