Réseaux de neurones en programmation fonctionnelle avec application à l'aide à la preuve.

1 Équipe et contacts

- Équipe Cosynus, LIX, École Polytechnique
- Encadrant principal: Jérémy Dubut jeremy.dubut@polytechnique.edu
- Co-encadrant: Samuel Mimram samuel.mimram@polytechnique.edu

Le stage s'effectuera au LIX à Palaiseau (Ile-de-France).

2 Mots clés

Ocaml, réseaux de neurones récurrents, théorie des types dépendants, génération de preuves

3 Contexte

Ces dernières années, l'apprentissage automatique (ML) a suscité un vif intérêt en tant qu'outil puissant pour résoudre des problèmes complexes dans divers domaines, notamment le raisonnement automatisé [9], la vérification formelle [13] et la satisfiabilité modulo des théories [12]. Bien que les avantages des méthodes d'apprentissage automatique dans ces domaines soient largement reconnus, leur intégration aux systèmes logiques traditionnels, tels que les assistants de preuve, reste un défi encore peu exploré, bien qu'il suscite un intérêt croissant au sein de plusieurs communautés [2].

Ce stage propose d'examiner comment la programmation fonctionnelle (FP), et en particulier l'utilisation du langage OCaml, peut offrir un cadre structuré et général pour concevoir un environnement expérimental (sandbox) permettant l'application et l'expérimentation des techniques de ML dans le contexte de la démonstration de théorèmes.

La programmation fonctionnelle est depuis longtemps reconnue pour sa capacité à modéliser des systèmes complexes, ce qui la rend particulièrement adaptée à la formalisation de structures et d'algorithmes mathématiques de manière claire et expressive. Il n'est donc pas surprenant que la majorité des assistants de preuve existants soient implémentés dans un langage fonctionnel [7].

OCaml, un langage de programmation fonctionnel statiquement typé, constitue un environnement puissant pour la construction de systèmes complexes nécessitant un haut niveau d'abstraction, de modularité et de performance. Dans le contexte de la démonstration de théorèmes, son riche système de types, combiné à des fonctionnalités avancées telles que les types algébriques, les fonctions d'ordre supérieur et le filtrage par motif, en fait un candidat idéal pour développer un cadre robuste capable de prendre en charge à la fois les approches logiques classiques et les méthodes basées sur l'apprentissage automatique. Voir par exemple les divers projets et bibliothèques Ocaml: [1, 3, 6, 5].

Les récents progrès en apprentissage automatique, notamment en apprentissage par renforcement, ouvrent la voie à l'amélioration des assistants de preuve traditionnels en suggérant automatiquement des tactiques pertinentes, en recommandant des étapes de démonstration, voire en générant des preuves de manière autonome [11, 8, 10, 14, 15].

Dans un cadre à types dépendants, la génération automatique de preuves représente un défi particulier en raison de l'interdépendance étroite entre les types et les termes, où les types peuvent dépendre des valeurs. Construire un arbre de preuve dans un tel contexte implique d'enchaîner des étapes successives qui garantissent non seulement la validité logique, mais aussi le respect des contraintes de dépendance imposées par les types. Contrairement aux approches

classiques basées sur des tactiques prédéfinies, l'objectif ici est d'engendrer automatiquement une structure de preuve cohérente, où chaque étape s'inscrit dans les contraintes définies par les types dépendants. Cette tâche s'apparente à la génération de séquences cohérentes par les modèles de langage, mais avec la complexité supplémentaire d'assurer la compatibilité stricte de chaque terme avec son type dépendant.

4 Objectifs

Plus concrètement:

1. Prendre en main les méthodes d'apprentissage utilisées dans la littérature en particulier pour des tâches d'aide à la preuve (recommandation de tactique, génération de preuve, ...)
2. Sur les bases de [4], implémenter une variété de réseaux de neurones, en particulier des réseaux récurrents et diverses variantes essentielles aux expérimentations du point 3. Le tout doit former une bibliothèque Ocaml permettant de construire des réseaux de manière générale et structurée.
3. Expérimenter la flexibilité et l'efficacité de cette bibliothèque sur de petites tâches d'apprentissage dans le domaine de l'aide à la preuve, en particulier dans un cadre de types dépendants, tout d'abord avec des méthodes connues (par exemple, pour de la recommendation de tactiques), puis en développant des méthodes plus spécifiques pour la génération de preuves dans des systèmes logiques simples (logique de combinateurs, lambda-pi calcul, ...).
4. Suivant les affinités du stagiaire, d'autres utilisations de cette bibliothèque sont envisageables.

5 Compétences requises (ou non)

- (fortement recommandée) Programmation fonctionnelle, et particulièrement en Ocaml
- (mieux mais pas nécessaire) Bases en algorithmes d'apprentissage et réseaux de neurones (en particulier réseaux récurrents, backpropagation, ...)
- (recommandée) Bases en théorie des types

References

[1] `https://ocaml.xyz/`.

[2] `https://europroofnet.github.io/wg5/`, last modified: Jan 2025.

[3] `https://github.com/ahrefs/ocannl`, last modified: Jan. 2025.

[4] `https://github.com/smimram/ocaml-backprop`, last modified: Jan. 2025.

[5] `https://github.com/LaurentMazare/tensorflow-ocaml`, last modified: Jul. 2019.

[6] `https://github.com/janestreet/torch`, last modified: Nov. 2024.

[7] `https://en.wikipedia.org/wiki/Proof_assistant#Comparison`, last modified: Oct. 2024.

[8] K. Bansal, S. Loos, M. Rabe, C. Szegedy, and S. Wilcox. Holist: An environment for machine learning of higher order logic theorem proving. In *International Conference on Machine Learning*, pages 454–463. PMLR, 2019.

[9] L. Blaauwbroek, D. M. Cerna, T. Gauthier, J. Jakubův, C. Kaliszyk, M. Suda, and J. Urban. Learning Guided Automated Reasoning: A Brief Survey. In *Logics and Type Systems in Theory and Practice*, 2024.

[10] T. Gauthier, C. Kaliszyk, J. Urban, R. Kumar, and M. Norrish. Tactictoe: learning to prove with tactics. *Journal of Automated Reasoning*, 65(2):257–286, 2021.

[11] D. Huang, P. Dhariwal, D. Song, and I. Sutskever. Gamepad: A learning environment for theorem proving. *preprint arXiv:1806.00608*, 2018.

[12] D. Selsam and N. Bjørner. Guiding high-performance SAT solvers with unsat-core predictions. In *Theory and Applications of Satisfiability Testing–SAT 2019: 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9–12, 2019, Proceedings 22*, pages 336–353, 2019.

[13] R. Sharma, S. Gupta, B. Hariharan, A. Aiken, and A. V. Nori. Verification as Learning Geometric Concepts. In *20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2012, Proceedings*, 2013.

[14] K. Yang and J. Deng. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning*, pages 6984–6994. PMLR, 2019.

[15] K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. J. Prenger, and A. Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36, 2024.