Abstract Interpretation of Automatic Differentiation

Despite the fact that Automatic Differentiation (AD) underlies much of machine learning, scientific computing and graphics, static analyses of AD code has been limited. In this position paper, we argue why AD code needs precise, general and scalable abstract interpretation and why this abstract interpretation should leverage the specific structure inherent to AD. Lastly, we demonstrate a new use of abstract interpretation of AD to analyze Coordinate MLPs from Graphics, attaining orders of magnitude more precision than prior work.

1 INTRODUCTION

Automatic Differentiation (AD) lies at the heart of many areas of Computer Science. Derivatives computed by AD form the backbone of Machine learning, as well as much of Scientific Computing [Griewank and Walther 2008] and Graphics [Bangaru et al. 2021]. Despite the ubiquity of AD computations, there is surprisingly limited work on analyzing formal properties defined over these derivatives. Nevertheless, there are many applications that require formal reasoning about derivatives. For example, monotonicity (which requires that all derivatives are either strictly positive or strictly negative) has been used as a metric for establishing algorithmic fairness [Shi et al. 2022]. Similarly, for optimization, previous work [Deussen 2021] needed an analysis that could certify a function was convex in a given input region using the second derivative. Similarly, [Ramasinghe and Lucey 2022] showed the need to bound local Lipschitz constants for reasoning about representational power of DNNs in Computer Graphics. In light of these needs and others, we argue that developers need a unified framework for supporting all of these AD program analyses. In this position paper, we argue why Abstract Interpretation is the most natural program analysis technique for providing, general, precise and scalable reasoning about Automatic Differentiation.

2 ABSTRACT INTERPRETATION OF AUTOMATIC DIFFERENTIATION

Abstract Interpretation is a program analysis technique that over-approximates the set of concrete program executions to prove useful program invariants [Cousot and Cousot 1977]. Building upon this idea, Abstract Interpretation of Automatic Differentiation aims to prove numeric invariants about the derivatives computed by AD. Numeric invariants such as lower and upper bounds on derivatives can be directly used to prove various properties such as monotonicity, convexity, Lipschitz robustness, among others. Since AD code is ultimately just a composition of primitive numeric operations (e.g., +, -, log, exp, etc.), one can use standard numeric abstract domains like Zonotopes [Ghorbal et al. 2009] or Polyhedra [Singh et al. 2019]. However, despite the seeming simplicity in applying existing numerical abstract interpretation literature to AD, multiple challenges arise.

Precision A direct application of Abstract Interpretation to AD can unfortunately lead to imprecise bounds. This issue arises because of the inherent non-linearity of derivative computations. We argue that an abstract interpretation should be designed with this fact in mind as in [Laurel et al. 2023], and should exploit the inherent structure of the chain rule, product rule and quotient rule in order to group multiple non-linear operations together to improve precision.

Generality. We argue that abstract interpretation of AD should simultaneously be general. However, to obtain this generality the analysis must support higher-order AD as well as support programs with non-differentiabilities. For the former, we argue that abstract interpretation of higher-order AD should follow a co-design of both the abstract and concrete semantics. We argue that one should first design a concrete semantics that lends itself to precise abstract interpretation. In particular,

Author's address:

2023. XXXX-XXXX/2023/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

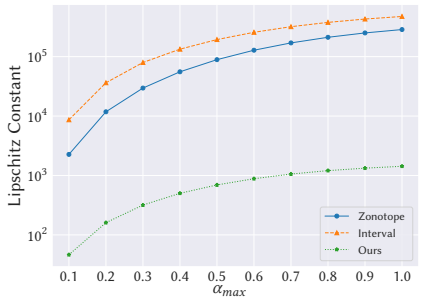

Fig. 1. Lipschitz bounds of a CMLP under varying perturbations over 100 coordinates in the 2D image space. This figure presents the average upper bounds on the local Lipschitz constants with respect to different α_{max} for the interval AD, zonotope AD, and our method.

prior work [Laurel et al. 2022b] has shown how to use variable sharing, whereby common subexpressions are eliminated across higher-derivatives in order to better capture data dependencies between these derivatives. We argue that such an approach is necessary to improve precision and allow developers to easily reason about their code.

Non-differentiability is another obstacle in the way of abstractly interpreting AD. To address this challenge, we also argue for an abstract semantics based upon the Clarke Generalized Jacobian [Laurel et al. 2022a]. The Clarke Jacobian is defined as:

$$\partial_c(f, \mathbf{x_0}) = \mathbf{co}\{\lim_{j\to\infty} \mathbf{J}(f, \mathbf{x}_j) : \lim_{j\to\infty} \mathbf{x}_j = \mathbf{x_0} \; and \; \mathbf{x}_j \notin S \text{ for all } j \in \mathbb{N}\} \tag{1}$$

The key benefit of the Clarke Jacobian is that it is well-defined even at points of non-differentiability, such as the ReLU at the origin. Hence a semantics defined over the Clarke Jacobian allows one to soundly reason about branches which can introduce non-differentiablities.

Scalability Since derivative computations typically have 2-5x more operations than the original function being differentiated [Griewank and Walther 2008], scalability is a primary concern. Given that Convolutional Neural Networks represent a popular use case, this challenge often arises in applied Machine learning settings. However prior work [Laurel et al. 2023] has been able to obtain scalability by developing analyses that can make use of tensor-level operations, and thus leverage GPU parallelism. We argue that these insights are a necessity for any AD static analysis.

3 FUTURE DIRECTIONS: ABSTRACT INTERPRETATION OF AD FOR GRAPHICS

In Graphics, Coordinate Multilayer Perceptrons (CMLPs) have emerged as a popular model for representing images and scenes. By learning a function that maps coordinates (e.g. x, y, z) to color values (e.g. R,G,B), these networks learn an implicit representation of 2D and 3D signals and thus are more compact than many traditional representations of images and surfaces. Recent work [Ramasinghe and Lucey 2022] has even established a unifying theoretical framework from which to understand the expressive power of CMLPs. They argue that the local Lipschitz constant around an input coordinate corresponds to the network's ability to represent visual phenomena like edges.

Motivated by their insight, we apply AD static analyzers ([Laurel et al. 2023]) to CMLPs for the first time. Specifically, we evaluate Lipschitz bounds of a CMLP under varying perturbations over 100 coordinates in the 2D image space. Our results are shown in Fig. 1. The local Lipschitz constant bounds produced by our technique are between 6-200× more precise than an interval-based or Zonotope-based abstract interpretation of AD.

4 CONCLUSION

Automatic Differentiation is here to stay, hence it is imperative to develop static analyses for derivatives. We argue that combining Abstract Interpretation with domain specific insights from AD is the best path forward to obtaining precise, general and scalable analyses. Additionally we show how these analyses can be adapted to Computer Graphics to obtain significant improvements.

REFERENCES

Sai Praveen Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically differentiating parametric discontinuities. *ACM Transactions on Graphics (TOG)* 40, 4 (2021).

Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*.

Jens Deussen. 2021. *Global Derivatives*. Ph. D. Dissertation.

Khalil Ghorbal, Eric Goubault, and Sylvie Putot. 2009. The zonotope abstract domain taylor1+. In *International Conference on Computer Aided Verification*. 627–633.

Andreas Griewank and Andrea Walther. 2008. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM.

Jacob Laurel, Siyuan Brant Qian, Gagandeep Singh, and Sasa Misailovic. 2023. Synthesizing precise static analyzers for automatic differentiation. *Proceedings of the ACM on Programming Languages* 7, OOPSLA2 (2023), 1964–1992.

Jacob Laurel, Rem Yang, Gagandeep Singh, and Sasa Misailovic. 2022a. A Dual Number Abstraction for Static Analysis of Clarke Jacobians. *Proceedings of the ACM on Programming Languages* POPL (2022), 1–30.

Jacob Laurel, Rem Yang, Shubham Ugare, Robert Nagel, Gagandeep Singh, and Sasa Misailovic. 2022b. A general construction for abstract interpretation of higher-order automatic differentiation. *Proceedings of the ACM on Programming Languages* 6, OOPSLA2 (2022), 1007–1035.

Sameera Ramasinghe and Simon Lucey. 2022. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In *European Conference on Computer Vision*. Springer, 142–158.

Zhouxing Shi, Yihan Wang, Huan Zhang, Zico Kolter, and Cho-Jui Hsieh. 2022. Efficiently Computing Local Lipschitz Constants of Neural Networks via Bound Propagation. In *Advances in Neural Information Processing Systems*.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30.