

JET FUNCTORS AND WEIL ALGEBRAS IN AUTOMATIC DIFFERENTIATION: A GEOMETRIC ANALYSIS

AMANDIP SANGHA

The Climate and Environmental Research Institute NILU, Norway

ABSTRACT. We present a differential-geometric formulation of automatic differentiation (AD) based on jet functors and Weil algebras. In this framework, forward- and reverse-mode differentiation arise naturally as pushforward and cotangent pullback, while higher-order differentiation corresponds to evaluation in a Weil algebra. This construction provides a unified, coordinate-free view of derivative propagation and clarifies the algebraic structure underlying AD. All results are realized in modern JAX code, where the Weil-mode formulation computes all mixed derivatives in a single forward pass with cost linear in the algebra dimension. The resulting implementation achieves algebraically exact and numerically stable differentiation with predictable scaling, demonstrating that geometric abstraction can yield more efficient and transparent computational differentiation systems. Code is available at <https://git.nilu.no/geometric-ad/jet-weil-ad>.

1. INTRODUCTION

Automatic differentiation (AD) provides an exact and efficient means of computing derivatives, yet its geometric interpretation is often underemphasized. This paper situates AD within a differential-geometric framework where forward- and reverse-mode differentiation arise naturally from the action of jet and Weil functors on smooth manifolds. In this formulation, reverse-mode AD corresponds to cotangent pullback, while higher-order (Taylor-mode) differentiation is expressed as evaluation in a Weil algebra, giving a coordinate-free account of derivative propagation.

This geometric viewpoint leads directly to more efficient computational implementations. All results are realized in modern JAX code, yielding algebraically exact derivative propagation and controlled numerical error. By exploiting the tensorized structure of Weil algebras, the approach computes all mixed derivatives in a single forward pass with cost linear in the algebra dimension, avoiding the combinatorial blow-up of nested JVP/VJP schedules. The resulting code provides concise, structure-preserving differentiation with predictable scaling and improved efficiency.

2. RELATED WORK

Prior research has explored the categorical and geometric foundations of automatic differentiation (AD) from several complementary angles. Elliott [1] and Fong, Spivak, and Tuyéras [2] introduced compositional and functorial accounts of reverse-mode AD, framing backpropagation as a cotangent pullback and clarifying its invariance under smooth reparametrizations. Baydin et al. [3] provided a comprehensive survey of AD algorithms and implementations, motivating the search for unified theoretical formalisms. Betancourt [4] extended these ideas to higher-order differentiation through jets, linking AD to the geometry of smooth manifolds. Giles [5] and Fike and Alonso [6] analyzed numerical stability and computational scaling of higher-order derivatives. Our work builds on these foundations by formulating reverse-, forward-, and higher-order AD as instances of functorial constructions on jet and Weil bundles, emphasizing both theoretical coherence and algorithmic efficiency. Prior geometric treatments of automatic differentiation, such as [4], identify forward- and

E-mail address: asan@nilu.no.

reverse-mode differentiation with pushforward and pullback on tangent and cotangent bundles. The present formulation extends this perspective by expressing higher-order differentiation as evaluation in a Weil algebra, yielding a functorial and computationally efficient realization implemented in modern JAX.

3. GEOMETRIC FOUNDATIONS

We summarize the geometric constructions required for our results: jet spaces and bundles, pushforwards and pullbacks, and Weil algebras. These notions provide the formal setting for automatic differentiation.

3.1. Smooth manifolds and differentials. Let M, N be finite-dimensional C^∞ manifolds. For a smooth map $f : M \rightarrow N$ and $x \in M$, the *differential* at x is

$$df_x : T_x M \longrightarrow T_{f(x)} N,$$

a linear map between tangent spaces. Its dual (cotangent) map is

$$(df_x)^* : T_{f(x)}^* N \longrightarrow T_x^* M,$$

called the *cotangent pullback* [7]. It satisfies contravariant functoriality:

$$(d(g \circ f)_x)^* = df_x^* \circ dg_{f(x)}^*.$$

3.2. Pushforward, pullback, and functoriality. For a smooth map $f : M \rightarrow N$ and $x \in M$:

- The *pushforward* transports tangent vectors forward:

$$df_x : T_x M \longrightarrow T_{f(x)} N, \quad v \mapsto df_x(v).$$

In local coordinates $(x^1, \dots, x^n)$ on M and $(y^1, \dots, y^m)$ on N , this is represented by the Jacobian matrix [7]

$$J_f(x) = \left[\frac{\partial f^i}{\partial x^j}(x) \right]_{i=1, \dots, m}^{j=1, \dots, n},$$

and the pushforward acts as

$$df_x(v) = J_f(x) v.$$

- The *pullback* transports covectors backward:

$$(df_x)^* : T_{f(x)}^* N \longrightarrow T_x^* M, \quad \omega \mapsto \omega \circ df_x.$$

In coordinates, for $\omega = (\omega_1, \dots, \omega_m) \in T_{f(x)}^* N$, the pullback acts by matrix transpose [1]:

$$(df_x)^*(\omega) = J_f(x)^\top \omega.$$

Automatic differentiation: forward and reverse. Automatic differentiation (AD) computes derivatives of programs by applying the chain rule to a composition of elementary primitives at machine precision and without finite differencing or symbolic expansion [1, 3]. A *program* in AD is a finite acyclic computational graph of differentiable primitives (basic operations or functions such as addition, multiplication, or smooth elementwise functions with known derivatives), mapping inputs $x \in \mathbb{R}^n$ to outputs $y \in \mathbb{R}^m$. Its semantics is the function obtained by evaluating nodes in topological order (an ordering of nodes so that every node appears after all its dependencies). A *straight-line program* (SLP) is a program with no branches or loops, represented by sequential assignments $u_k = \phi_k(u_{i_1}, \dots, u_{i_{r_k}})$ with $i_j < k$; AD then applies local derivative propagation rules (forward mode for directional derivatives and reverse mode for gradients) along this sequence to compute derivatives efficiently.

Throughout, let

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad y = f(x), \quad J_f(x) = \left[\frac{\partial f_i}{\partial x_j}(x) \right]_{i=1, \dots, m}^{j=1, \dots, n},$$

and recall from §3.2 that the differential $df_x : T_x\mathbb{R}^n \rightarrow T_{f(x)}\mathbb{R}^m$ (pushforward) and its dual $(df_x)^* : T_{f(x)}^*\mathbb{R}^m \rightarrow T_x^*\mathbb{R}^n$ (cotangent pullback) encode first-order variation geometrically [7, 8].

Forward mode (pushforward/JVP).. JVP stands for *Jacobian–Vector Product*, i.e., multiplying the Jacobian $J_f(x)$ by a tangent vector v to propagate directional derivatives. Given a seed (tangent) direction $v \in \mathbb{R}^n$, the directional derivative of f at x in direction v is

$$\dot{y} = df_x(v) = J_f(x)v,$$

i.e. JVP [3, 5]. Operationally, each intermediate scalar u is paired with its tangent $\dot{u}$, and elementary rules implement the chain rule:

$$\begin{aligned} u = a + b &\Rightarrow \dot{u} = \dot{a} + \dot{b}, \\ u = a \cdot b &\Rightarrow \dot{u} = \dot{a}b + a\dot{b}, \\ u = \varphi(a) &\Rightarrow \dot{u} = \varphi'(a)\dot{a} \quad (\varphi \in C^1). \end{aligned}$$

In the manifold terms of §3.1-3.2, forward mode computes the pushforward df_x [7, 8].

Reverse mode (pullback/VJP).. VJP stands for *Vector–Jacobian Product*, i.e., multiplying the transpose $J_f(x)^\top$ by a "seed" covector $\omega = \nabla\ell(y)$ to propagate gradients backward. Here $\nabla\ell(y)$ denotes the gradient of a scalar function $\ell : \mathbb{R}^m \rightarrow \mathbb{R}$,

$$\nabla\ell(y) = \left(\frac{\partial\ell}{\partial y_1}, \dots, \frac{\partial\ell}{\partial y_m} \right)^\top.$$

Given a seed covector $\omega = \nabla\ell(y) \in \mathbb{R}^m$, reverse mode computes the Vector-Jacobian Product (VJP)

$$\bar{x} = (df_x)^*(\omega) = J_f(x)^\top \omega,$$

characterized by the dual pairing (evaluation map) $\langle \cdot, \cdot \rangle_x : T_x^*\mathbb{R}^n \times T_x\mathbb{R}^n \longrightarrow \mathbb{R}$,

$$\langle (df_x)^*\omega, v \rangle = \langle \omega, df_x(v) \rangle \quad \forall v \in T_x\mathbb{R}^n,$$

i.e. the cotangent pullback along f [7, 8]. The corresponding adjoint rules apply along a reverse sweep:

$$\begin{aligned} u = a + b &\Rightarrow \bar{a} += \bar{u}, \quad \bar{b} += \bar{u}, \\ u = a \cdot b &\Rightarrow \bar{a} += b\bar{u}, \quad \bar{b} += a\bar{u}, \\ u = \varphi(a) &\Rightarrow \bar{a} += \varphi'(a)\bar{u}. \end{aligned}$$

For $m = 1$ (scalar output) this yields the full gradient $\nabla_x f(x)$ in time within a small constant factor of a primal evaluation, which underlies backpropagation in deep learning [3, 5]. Categorically, reverse mode is the contravariant action (pullback) on covelocities—"backprop as functor" [2, 1].

Chain rule in both modes. For a composition $g \circ f$,

$$d(g \circ f)_x = dg_{f(x)} \circ df_x, \quad (d(g \circ f)_x)^* = (df_x)^* \circ (dg_{f(x)})^*,$$

which in coordinates gives

$$J_{g \circ f}(x) = J_g(f(x))J_f(x), \quad \nabla_x(\ell \circ f)(x) = J_f(x)^\top \nabla\ell(f(x)).$$

These are precisely the functorial identities for pushforwards and cotangent pullbacks invoked in our geometric framework (§3.3–§3.3), providing a coordinate-free account of JVP/VJP and backpropagation [7, 8, 2]. In the manifold terms of §3.1-3.2, reverse mode computes the cotangent pullback $(df_x)^*$.

3.3. Jets, Weil algebras, and functorial AD. For $k \geq 1$, two smooth maps $f, g : M \rightarrow N$ have the same k -jet at x if all partial derivatives up to order k agree in local charts. The corresponding equivalence class is denoted $j_x^k f$. The k -jet bundle $J^k(M, N)$ is the bundle of all such jets, with projection $\pi^k : J^k(M, N) \rightarrow M$. The jet prolongation of f is the section

$$j^k f : M \longrightarrow J^k(M, N), \quad x \longmapsto j_x^k f.$$

We denote by **Man** the category whose objects are smooth finite-dimensional manifolds and whose morphisms are smooth maps. Its opposite category **Man**^{op} has the same objects but with arrows reversed.

The assignment $(M, N, f) \mapsto J^k(M, N)$ extends (for a fixed manifold N) to a *functor*

$$J^k(-, N) : \mathbf{Man} \longrightarrow \mathbf{Man},$$

or, more precisely a bifunctor $J^k : \mathbf{Man}^{\text{op}} \times \mathbf{Man} \longrightarrow \mathbf{Man}$, but we will only consider $J^k = J^k(-, N)$ here. This functor acts on maps by

$$J^k(f)(j_x^k \phi) = j_x^k(f \circ \phi),$$

and satisfies the naturality property [7, 8]:

$$J^k(g \circ f) = J^k(g) \circ J^k(f).$$

In this sense, k -jets encode the truncated Taylor expansion of smooth maps up to order k , independently of any choice of coordinates. Indeed, in local coordinates (x^i) on M and (y^α) on N , the k -jet of f at x is determined by the collection of partial derivatives

$$\left\{ \frac{\partial^{|\beta|} f^\alpha}{\partial x^\beta}(x) \mid 1 \leq \alpha \leq \dim N, |\beta| \leq k \right\},$$

which represents the coefficients of the truncated Taylor polynomial of f at x . The functoriality of J^k expresses the compatibility of these derivatives with composition: if $f : M \rightarrow N$ and $g : N \rightarrow P$, then in local coordinates

$$\frac{\partial^{|\gamma|} (g^\lambda \circ f)}{\partial x^\gamma}(x) = F_\gamma \left(\frac{\partial^{|\beta|} f^\alpha}{\partial x^\beta}(x), \frac{\partial^{|\delta|} g^\lambda}{\partial y^\delta}(f(x)) \right),$$

where F_γ is the universal polynomial given by the multivariate chain rule (Faà di Bruno formula, [9]). Thus, J^k packages all partial derivatives of order $\leq k$ into a coordinate-free object whose transformation law under composition is precisely the functorial identity above.

Weil algebras and Weil functors. A *Weil algebra* W [8, 10] is a finite-dimensional local $\mathbb{R}$ -algebra with maximal ideal $\mathfrak{m}$ such that $\mathfrak{m}^{k+1} = 0$ for some k . Examples:

$$\mathbb{R}[\varepsilon]/(\varepsilon^2), \quad \mathbb{R}[\varepsilon]/(\varepsilon^{k+1}), \quad \bigotimes_{j=1}^p \mathbb{R}[\varepsilon_j]/(\varepsilon_j^{\rho_j+1}).$$

The *Weil functor* T_W assigns to each manifold M the space $T_W M$ of W -points of M , and to each smooth map $f : M \rightarrow N$ the lifted map $T_W f : T_W M \rightarrow T_W N$. Evaluating f on T_W propagates truncated Taylor expansions coefficientwise, enabling higher-order AD in one pass.

Classically, the k -jet bundle $J^k(M, N)$ collects all Taylor expansions of smooth maps $f : M \rightarrow N$ up to order k between smooth manifolds M and N , and composition of maps satisfies $J^k(g \circ f) = J^k(g) \circ J^k(f)$ [7, 8]. A Weil algebra W with $\mathfrak{m}^{k+1} = 0$ provides an algebraic model of these truncated jets: the space $T_W M$ of W -points of M corresponds to the fiber of $J^k(M, \mathbb{R}^n)$ at x . Explicitly, one may take

$$W = \mathbb{R}[\varepsilon_1, \dots, \varepsilon_p]/(\varepsilon_1, \dots, \varepsilon_p)^{k+1},$$

so that its maximal ideal is $\mathfrak{m} = \langle \varepsilon_1, \dots, \varepsilon_p \rangle$ with $\mathfrak{m}^{k+1} = 0$. Every element $w \in W$ has a unique expansion

$$w = a_0 + \sum_{1 \leq |\alpha| \leq k} a_\alpha \varepsilon^\alpha, \quad \varepsilon^\alpha := \varepsilon_1^{\alpha_1} \cdots \varepsilon_p^{\alpha_p},$$

with coefficients $a_\alpha \in \mathbb{R}$. A W -point of a manifold M near $x \in M$ is then represented in local coordinates by

$$x_W = x + \sum_{1 \leq |\alpha| \leq k} h_\alpha \varepsilon^\alpha, \quad h_\alpha \in T_x M,$$

encoding all infinitesimal displacements up to order k along the formal nilpotent directions ε_i . Evaluating a smooth map $f: M \rightarrow N$ on x_W yields

$$f(x_W) = f(x) + \sum_{1 \leq |\alpha| \leq k} \frac{1}{\alpha!} D^\alpha f(x) [h_\alpha] \varepsilon^\alpha,$$

so that the coefficients of ε^α are precisely the partial derivatives of f at x . Formally, there is a canonical correspondence between W and the k -jet bundle:

$$T_W M \simeq J^k(M, \mathbb{R}^n).$$

Under this identification, a W -point

$$x_W = x + \sum_{1 \leq |\alpha| \leq k} h_\alpha \varepsilon^\alpha \quad \text{corresponds to} \quad j_x^k(\text{id}_M) \in J^k(M, \mathbb{R}^n),$$

and evaluating f on x_W recovers the same truncated Taylor expansion as the k -jet $j_x^k f$. Hence the Weil construction T_W can be viewed as an algebraic model of the geometric jet functor, and we work directly with W for analytic clarity and computational convenience. Indeed, throughout the paper we work directly with W rather than J^k , since all constructions, derivative exactness, and complexity bounds follow from the nilpotent algebra structure without requiring explicit jet-bundle machinery.

Definition 1 (Primitive operation). *A primitive is an elementary differentiable map $P: \mathbb{R}^r \rightarrow \mathbb{R}$ drawn from a fixed library $\mathcal{P}$ of basic operations (affine, bilinear, elementwise C^∞ , reductions, etc.). Every program f is represented as a straight-line composition of such primitives.*

Definition 2 (Lifted operation). *Given a primitive $P: \mathbb{R}^r \rightarrow \mathbb{R}$, its lift to a Weil algebra $W^{(\leq k)}$ is the map*

$$T_W P: W^r \longrightarrow W, \quad (x_1, \dots, x_r) \longmapsto P(x_1, \dots, x_r)$$

where the evaluation is performed coefficientwise on the truncated Taylor expansions of x_i . Thus each primitive acts on Weil elements instead of scalars, updating the full set of derivative coefficients in a single pass.

Definition 3 (Coefficients per scalar). *In the truncated Weil (or multivariate Taylor) lift of total order k over a p -dimensional seed subspace $U = \text{span}\{\varepsilon_1, \dots, \varepsilon_p\}$, each scalar variable $x_0 \in \mathbb{R}$ is replaced by its truncated formal expansion*

$$x \longmapsto x_0 + \sum_{|\alpha| \leq k} \varepsilon^\alpha x_\alpha, \quad \varepsilon^\alpha := \varepsilon_1^{\alpha_1} \cdots \varepsilon_p^{\alpha_p}.$$

The total number of distinct monomials ε^α with total degree $|\alpha| \leq k$ is

$$C(p, k) := \sum_{\ell=0}^k \binom{p+\ell-1}{\ell} = \binom{p+k}{k}.$$

We call $C(p, k)$ the number of coefficients per scalar, since every scalar in the lifted program is represented by a vector of $C(p, k)$ Taylor coefficients corresponding to all mixed partial derivatives of total order $\leq k$ along the p seed directions.

3.4. AD primitives and their geometric meaning. For $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ at x , the *pushforward* df_x corresponds to the JVP $v \mapsto J_f(x)v$, and the *cotangent pullback* $(df_x)^*$ corresponds to the VJP $\omega \mapsto J_f(x)^\top \omega$. We use the latter in Theorem 1 and in the respective experiments to realize reverse-mode as cotangent pullback. Functorially, these correspond to df_x and $(df_x)^*$, respectively. Higher-order AD uses jet composition or Weil functor evaluation to propagate derivatives up to order k .

4. CORE THEOREMS OF THE GEOMETRIC AD FRAMEWORK

Theorem 1 (Backpropagation as cotangent pullback and its Weil extension). *Let $f : M \rightarrow N$ be a smooth map between manifolds and $\ell : N \rightarrow \mathbb{R}$ a smooth scalar loss function. Then reverse-mode differentiation (backpropagation) satisfies*

$$\nabla_x(\ell \circ f) = (df_x)^*(d\ell_{f(x)}),$$

that is, the gradient of the composed loss is obtained by pulling back the covector $d\ell_{f(x)} \in T_{f(x)}^*N$ along the cotangent map $(df_x)^* : T_{f(x)}^*N \rightarrow T_x^*M$.

More generally, for any Weil algebra $W = \mathbb{R}[\varepsilon_1, \dots, \varepsilon_p]/(\varepsilon_1, \dots, \varepsilon_p)^{k+1}$ with maximal ideal $\mathfrak{m}$ satisfying $\mathfrak{m}^{k+1} = 0$, the lifted maps

$$T_W f : T_W M \rightarrow T_W N, \quad T_W \ell : T_W N \rightarrow T_W \mathbb{R} \cong W,$$

satisfy the Weil-level pullback identity

$$T_W^*(\ell \circ f) = T_W^* f \circ T_W^* \ell.$$

In coordinates, for a Weil-covector $\lambda_W = \sum_{0 \leq |\alpha| \leq k} \lambda_\alpha \varepsilon^\alpha \in T_W^*N$, we have

$$T_W^*(\ell \circ f)(\lambda_W) = \sum_{0 \leq |\alpha| \leq k} (D^\alpha f(x))^\top D^\alpha \ell(f(x)) \varepsilon^\alpha.$$

For $k = 1$ and $W = \mathbb{R}[\varepsilon]/(\varepsilon^2)$, this reduces to the classical cotangent pullback formula above, recovering the standard reverse-mode rule of first-order AD.

Proof. Coordinate-free formulation. For smooth maps $f : M \rightarrow N$ and $\ell : N \rightarrow \mathbb{R}$, the differential of the composition satisfies the chain rule

$$d(\ell \circ f)_x = d\ell_{f(x)} \circ df_x,$$

for each $x \in M$. Taking adjoints gives

$$(d(\ell \circ f)_x)^* = (df_x)^* \circ (d\ell_{f(x)})^*,$$

and since $d\ell_{f(x)}$ is scalar-valued, this yields the familiar backpropagation identity

$$\nabla_x(\ell \circ f) = (df_x)^*(d\ell_{f(x)}).$$

For the Weil-level extension, the same composition property holds for the lifted functors. Given a Weil algebra $W = \mathbb{R}[\varepsilon_1, \dots, \varepsilon_p]/(\varepsilon_1, \dots, \varepsilon_p)^{k+1}$, the differential prolongations

$$T_W f : T_W M \rightarrow T_W N, \quad T_W \ell : T_W N \rightarrow T_W \mathbb{R} \cong W$$

satisfy

$$T_W(\ell \circ f) = T_W \ell \circ T_W f.$$

Taking duals yields the cotangent composition law

$$T_W^*(\ell \circ f) = T_W^* f \circ T_W^* \ell,$$

which generalizes the usual chain rule for the cotangent pullback to all orders encoded in W . This identity is independent of coordinates and follows directly from the naturality of the Weil functor.

Coordinate-based formulation. Let (x^i) be local coordinates on M near x and (y^j) on N near $y = f(x)$. Write $f = (f^j)$ so that $y^j = f^j(x)$, and let $\ell = \ell(y^1, \dots, y^m)$.

First order (classical chain rule). The differential of $\ell \circ f$ at x is

$$d(\ell \circ f)_x = \sum_{j=1}^m \partial_{y^j} \ell(f(x)) df_x^j,$$

and the cotangent pullback of the covector $d\ell_{f(x)} = \sum_j \partial_{y^j} \ell(f(x)) dy^j$ is

$$(df_x)^*(d\ell_{f(x)}) = \sum_{j=1}^m \partial_{y^j} \ell(f(x)) (df_x^j)^*,$$

which yields the familiar identity $\nabla_x(\ell \circ f) = (df_x)^*(d\ell_{f(x)})$.

Weil level. Let $W = \mathbb{R}[\varepsilon_1, \dots, \varepsilon_p]/(\varepsilon)^{k+1}$ and choose a W -point $x_W \in T_W M$ with coordinate expression $x_W = x + \sum_{1 \leq |\alpha| \leq k} h_\alpha \varepsilon^\alpha$ (in local coordinates). Then

$$T_W f(x_W)^j = f^j(x) + \sum_{1 \leq |\alpha| \leq k} \frac{1}{\alpha!} D_x^\alpha f^j(x) [h_\alpha] \varepsilon^\alpha \quad (j = 1, \dots, m),$$

so the lifted output is $y_W = (T_W f(x_W)^1, \dots, T_W f(x_W)^m)$ in W^m . Applying ℓ in y -coordinates,

$$T_W \ell(y_W) = \ell(f(x)) + \sum_{1 \leq |\alpha| \leq k} \frac{1}{\alpha!} \sum_{|\beta| \leq |\alpha|} \partial_y^\beta \ell(f(x)) \mathcal{F}_{\beta, \alpha}(\{D_x^\gamma f(x) [h_\gamma]\}) \varepsilon^\alpha,$$

where $\partial_y^\beta \ell$ denotes the partials of ℓ in the y^j -coordinates, and $\mathcal{F}_{\beta, \alpha}$ are the multivariate Faà–di–Bruno polynomials (combinations of the jets $D_x^\gamma f$ producing the α -coefficient). Taking the cotangent pullback on a Weil-covector $\lambda_W = \sum_{0 \leq |\alpha| \leq k} \lambda_\alpha \varepsilon^\alpha \in T_W^* N$ gives, coefficientwise in the y^j -coordinates,

$$T_W^*(\ell \circ f)(\lambda_W) = \sum_{0 \leq |\alpha| \leq k} \left(\sum_{|\beta| \leq |\alpha|} (\partial_y^\beta \ell)(f(x)) \mathcal{F}_{\beta, \alpha}(\{D_x^\gamma f(x)\}) \right)^\top \lambda_\alpha \varepsilon^\alpha.$$

For $k = 1$, only the first-order terms remain and we recover

$$T_W^*(\ell \circ f)(\lambda) = \sum_j \partial_{y^j} \ell(f(x)) (Df^j(x))^\top \lambda \varepsilon,$$

which corresponds to the classical cotangent pullback in coordinates. $\square$

Example 1 (Illustration of the cotangent pullback and its Weil extension). Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ and $\ell : \mathbb{R}^2 \rightarrow \mathbb{R}$ be defined by

$$f(x_1, x_2) = (x_1 + x_2^2, e^{x_1}), \quad \ell(y_1, y_2) = y_1 + \frac{1}{2} y_2^2.$$

$$df_x = \begin{pmatrix} \partial_{x_1} f^1 & \partial_{x_2} f^1 \\ \partial_{x_1} f^2 & \partial_{x_2} f^2 \end{pmatrix} = \begin{pmatrix} 1 & 2x_2 \\ e^{x_1} & 0 \end{pmatrix}, \quad d\ell_{f(x)} = (\partial_{y_1} \ell \quad \partial_{y_2} \ell)_{y=f(x)} = (1 \quad e^{x_1}).$$

$$(df_x)^*(d\ell_{f(x)}) = (df_x)^\top \begin{pmatrix} 1 \\ e^{x_1} \end{pmatrix} = \begin{pmatrix} 1 & e^{x_1} \\ 2x_2 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ e^{x_1} \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + e^{x_1} \cdot e^{x_1} \\ 2x_2 \cdot 1 + 0 \cdot e^{x_1} \end{pmatrix} = \begin{pmatrix} 1 + e^{2x_1} \\ 2x_2 \end{pmatrix}.$$

Note that in the calculation above, $d\ell_{f(x)}$ is a covector written as a row; for the matrix product $(df_x)^\top (d\ell_{f(x)})^\top$ we use its column representation corresponding to the adjoint $(df_x)^*$ acting on $d\ell_{f(x)}$.

Now, on the other hand,

$$\nabla_x(\ell \circ f) = \begin{pmatrix} \partial_{x_1}(\ell \circ f) \\ \partial_{x_2}(\ell \circ f) \end{pmatrix} = \begin{pmatrix} \partial_{x_1}(x_1 + x_2^2 + \frac{1}{2}e^{2x_1}) \\ \partial_{x_2}(x_1 + x_2^2 + \frac{1}{2}e^{2x_1}) \end{pmatrix} = \begin{pmatrix} 1 + e^{2x_1} \\ 2x_2 \end{pmatrix}.$$

Thus $(df_x)^*(d\ell_{f(x)}) = \nabla_x(\ell \circ f)$.

Now take the Weil algebra $W = \mathbb{R}[\varepsilon_1]/(\varepsilon_1^3)$ corresponding to univariate jets up to order $k = 2$. A W -point of $\mathbb{R}^2$ is

$$x_W = (x_1 + h_1\varepsilon_1 + h_2\varepsilon_1^2, x_2 + k_1\varepsilon_1 + k_2\varepsilon_1^2).$$

Expanding $T_W f(x_W)$ gives

$$\begin{aligned} T_W f(x_W) &= (f_1, f_2) = (x_1 + x_2^2, e^{x_1}) + (h_1 + 2x_2k_1, e^{x_1}h_1)\varepsilon_1 \\ &\quad + (h_2 + 2x_2k_2 + k_1^2, e^{x_1}(h_2 + \frac{1}{2}h_1^2))\varepsilon_1^2. \end{aligned}$$

Applying ℓ gives

$$\begin{aligned} T_W(\ell \circ f)(x_W) &= \ell(f(x)) + D\ell(f(x)) [Df(x)(h_1, k_1)] \varepsilon_1 \\ &\quad + \frac{1}{2} D^2\ell(f(x)) [Df(x)(h_1, k_1), Df(x)(h_1, k_1)] \varepsilon_1^2 + \cdots \end{aligned}$$

Taking the Weil-cotangent pullback $(T_W f)^*$ and acting on $\lambda_W = d\ell_{f(x)} + \lambda_1\varepsilon_1 + \lambda_2\varepsilon_1^2$ gives

$$T_W^*(\ell \circ f)(\lambda_W) = (Df(x))^\top D\ell(f(x)) + (D^2f(x))^\top D^2\ell(f(x)) \varepsilon_1 + \cdots,$$

which reproduces the coefficientwise higher-order pullback structure of Theorem 1. For $k = 1$, the ε_1^2 terms vanish and the expression reduces to the classical cotangent pullback computed above.

Theorem 2 (Exactness of Weil-mode evaluation). *Let W be a Weil algebra with maximal ideal $\mathfrak{m}$ satisfying $\mathfrak{m}^{k+1} = 0$. For any smooth map $f \in C^{k+1}(U, \mathbb{R}^m)$ and $x \in U$, the lifted map*

$$T_W f : T_W U \longrightarrow T_W \mathbb{R}^m$$

computes all derivatives of f up to order k at x exactly as the coefficients of the truncated Taylor expansion.

Proof. Since W is a local $\mathbb{R}$ -algebra with $\mathfrak{m}^{k+1} = 0$, every element $w \in W$ can be written as

$$w = a_0 + \sum_{|\alpha| \leq k} a_\alpha \varepsilon^\alpha, \quad \varepsilon^\alpha = \varepsilon_1^{\alpha_1} \cdots \varepsilon_r^{\alpha_r},$$

where $\varepsilon_i \in \mathfrak{m}$ are nilpotent generators and $a_\alpha \in \mathbb{R}$. Nilpotency ensures $\varepsilon^\alpha = 0$ whenever $|\alpha| > k$. Each point of $T_W U$ corresponds to a morphism $\text{Spec}(W) \rightarrow U$, which in local coordinates $(x^1, \dots, x^n)$ is represented by

$$x_W = x + \sum_{|\alpha| \leq k} h_\alpha \varepsilon^\alpha, \quad h_\alpha \in \mathbb{R}^n.$$

This encodes a formal Taylor expansion truncated at order k . Applying f to x_W and expanding in a multivariate Taylor series gives:

$$f(x_W) = f(x) + \sum_{|\alpha|=1}^k \frac{1}{\alpha!} D^\alpha f(x) [h_\alpha] \varepsilon^\alpha,$$

where $D^\alpha f(x)$ denotes the α -th derivative (multilinear map) of f at x . Because ε^α vanish for $|\alpha| > k$, the series truncates exactly at order k . The coefficients of ε^α in $f(x_W)$ are precisely the derivatives:

$$\text{Coefficient of } \varepsilon^\alpha = \frac{1}{\alpha!} D^\alpha f(x) [h_\alpha].$$

Thus, $T_W f$ computes $(f(x), Df(x), \dots, D^k f(x))$ algebraically, without numerical approximation or step-size tuning. Floating-point error propagates only through coefficient arithmetic, not through truncation, since truncation is enforced by $\mathfrak{m}^{k+1} = 0$. $\square$

Example 2 (Illustration of Theorem 2). *Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be $f(x) = \sin x$ and consider the Weil algebra $W = \mathbb{R}[\varepsilon]/(\varepsilon^3)$ corresponding to truncation at $k = 2$. A W -point near x is $x_W = x + h_1\varepsilon + h_2\varepsilon^2$. Then*

$$T_W f(x_W) = \sin(x + h_1\varepsilon + h_2\varepsilon^2) = \sin x + h_1 \cos x \varepsilon + (h_2 \cos x - \tfrac{1}{2}h_1^2 \sin x)\varepsilon^2.$$

The coefficients of ε and ε^2 are exactly the first and second derivatives of f at x evaluated along the seed directions h_1, h_2 :

$$\text{Coeff}[\varepsilon] = f'(x)h_1, \quad \text{Coeff}[\varepsilon^2] = \tfrac{1}{2}f''(x)h_1^2 + f'(x)h_2.$$

Thus $T_W f$ produces the truncated Taylor expansion $f(x_W) = f(x) + f'(x)h_1\varepsilon + \tfrac{1}{2}f''(x)h_1^2\varepsilon^2 + \dots$ exactly, demonstrating the algebraic exactness of Weil-mode evaluation.

Corollary 1 (Coefficient growth envelope). *Let $x \in \mathbb{R}^n$, and assume $f \in C^{k+1}(B_r(x), \mathbb{R}^m)$, and that its derivatives satisfy*

$$\|D^\ell f(z)\| \leq M_\ell \quad \text{for all } z \in B_r(x), \quad 0 \leq \ell \leq k+1,$$

where $B_r(x) = \{y \in \mathbb{R}^n \mid \|y - x\| < r\}$ denotes the open ball of radius r centered at x .

Then the Taylor coefficients of $T_W f$ at x obey

$$\|f_\alpha(x)\| \leq \frac{M_{|\alpha|}}{\alpha!}, \quad \text{for all multi-indices } \alpha \text{ with } |\alpha| \leq k,$$

and the truncated series is numerically stable for any step size $\rho < r$ with explicit tail bounds.

Proof. For $z \in B_r(x)$, write the Taylor expansion of f at x :

$$f(z) = \sum_{\ell=0}^k \sum_{|\alpha|=\ell} \frac{1}{\alpha!} D^\alpha f(x) [(z-x)^\alpha] + R_{k+1}(z),$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$, $\alpha! = \alpha_1! \cdots \alpha_n!$, and $|\alpha| = \alpha_1 + \dots + \alpha_n$. When evaluating f on $x_W = x + \sum_j \varepsilon_j h_j$ in T_W , the coefficient of ε^α in $f(x_W)$ is:

$$f_\alpha(x) = \frac{1}{\alpha!} D^\alpha f(x) [h_1^{\alpha_1}, \dots, h_n^{\alpha_n}],$$

where h_j are the seeded directions. For norm bounds, assume $\|h_j\| \leq 1$ (unit directions). By assumption, $\|D^\alpha f(x)\| \leq M_{|\alpha|}$ for all $|\alpha| \leq k$. Thus:

$$\|f_\alpha(x)\| \leq \frac{M_{|\alpha|}}{\alpha!} \cdot \|h_1^{\alpha_1}\| \cdots \|h_n^{\alpha_n}\| \leq \frac{M_{|\alpha|}}{\alpha!}.$$

For any $\rho < r$, the remainder term satisfies the standard Cauchy estimate:

$$\|R_{k+1}(z)\| \leq \frac{M_{k+1}}{(k+1)!} \|z - x\|^{k+1} \leq \frac{M_{k+1}}{(k+1)!} \rho^{k+1}.$$

Thus the truncated series is stable for $\rho < r$ with explicit control on the tail. Since truncation is enforced algebraically by $\mathfrak{m}^{k+1} = 0$, floating-point error propagates only through coefficient arithmetic, not through step-size selection. $\square$

Theorem 3 (Complexity of Tensorized Weil Algebras). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be C^k at $x \in \mathbb{R}^n$. Fix p input directions $v^{(1)}, \dots, v^{(p)} \in \mathbb{R}^n$ and per-direction truncation orders $\rho_1, \dots, \rho_p \in \mathbb{N}$ with $\rho_j \leq k$. Consider the tensorized Weil algebra*

$$W \cong \bigotimes_{j=1}^p \mathbb{R}[\varepsilon_j]/(\varepsilon_j^{\rho_j+1}), \quad \dim W = \prod_{j=1}^p (\rho_j + 1),$$

with commuting nilpotents $\varepsilon_i \varepsilon_j = \varepsilon_j \varepsilon_i$ and $\varepsilon_j^{\rho_j+1} = 0$. A single evaluation of f on T_W at the W -point

$$x_W := x + \sum_{j=1}^p \varepsilon_j v^{(j)}$$

produces all mixed directional derivatives of order $\leq k$ along the chosen directions, as the coefficients of the basis monomials $\varepsilon^\alpha := \varepsilon_1^{\alpha_1} \cdots \varepsilon_p^{\alpha_p}$ for all multi-indices $\alpha \in \mathbb{N}^p$ with $0 \leq \alpha_j \leq \rho_j$. Moreover, if the primal straight-line program for f uses Q scalar primitives, then the lifted evaluation uses $O(\dim W)$ coefficient arithmetic per primitive (hence time $O(\dim W) \cdot Q$ and memory $O(\dim W)$ per intermediate, with no adjoint tape).

Proof. By construction

$$W \cong \mathbb{R}[\varepsilon_1, \dots, \varepsilon_p] / (\varepsilon_1^{\rho_1+1}, \dots, \varepsilon_p^{\rho_p+1}),$$

with commuting variables $\varepsilon_i \varepsilon_j = \varepsilon_j \varepsilon_i$. Every $w \in W$ admits a unique expansion

$$w = \sum_{0 \leq \alpha_j \leq \rho_j} c_\alpha \varepsilon^\alpha, \quad \varepsilon^\alpha := \varepsilon_1^{\alpha_1} \cdots \varepsilon_p^{\alpha_p}, \quad c_\alpha \in \mathbb{R},$$

and the set $\{\varepsilon^\alpha : 0 \leq \alpha_j \leq \rho_j\}$ is an $\mathbb{R}$ -basis of W , hence $\dim W = \prod_{j=1}^p (\rho_j + 1)$. Define

$$x_W = x + \sum_{j=1}^p \varepsilon_j v^{(j)} \in (\mathbb{R}^n) \otimes W \cong T_W \mathbb{R}^n.$$

In coordinates, x_W has coefficient array supported on degree 0 (the basepoint x) and degree 1 (the first-order seeds $v^{(j)}$), with no higher-degree seeds required; powers of the ε_j generated inside W will encode higher orders automatically. Let $D^\ell f(x) : (\mathbb{R}^n)^\ell \rightarrow \mathbb{R}^m$ denote the symmetric ℓ -linear differential. Using the multivariate Taylor formula up to total order k and nilpotency to truncate:

$$\begin{aligned} f(x_W) &= \sum_{\ell=0}^k \frac{1}{\ell!} D^\ell f(x) \left[\underbrace{(x_W - x), \dots, (x_W - x)}_{\ell \text{ times}} \right] \\ &= \sum_{\ell=0}^k \frac{1}{\ell!} D^\ell f(x) \left[\sum_{j=1}^p \varepsilon_j v^{(j)}, \dots, \sum_{j=1}^p \varepsilon_j v^{(j)} \right]. \end{aligned}$$

Expanding the ℓ -fold multilinear form and using commutativity of the ε_j yields

$$\begin{aligned} f(x_W) &= \sum_{\ell=0}^k \frac{1}{\ell!} \sum_{j_1, \dots, j_\ell=1}^p \left(\varepsilon_{j_1} \cdots \varepsilon_{j_\ell} \right) D^\ell f(x) [v^{(j_1)}, \dots, v^{(j_\ell)}] \\ &= \sum_{\ell=0}^k \sum_{\substack{\alpha \in \mathbb{N}^p \\ |\alpha|=\ell}} \frac{1}{\ell!} \cdot \frac{\ell!}{\alpha_1! \cdots \alpha_p!} \varepsilon^\alpha D^\ell f(x) \left[\underbrace{v^{(1)}, \dots, v^{(1)}}_{\alpha_1}, \dots, \underbrace{v^{(p)}, \dots, v^{(p)}}_{\alpha_p} \right] \\ &= \sum_{0 \leq \alpha_j \leq \rho_j} \frac{1}{\alpha_1! \cdots \alpha_p!} \varepsilon^\alpha D^{|\alpha|} f(x) [v^{(1) \times \alpha_1}, \dots, v^{(p) \times \alpha_p}], \end{aligned}$$

where $|\alpha| := \alpha_1 + \cdots + \alpha_p$, and we used that each $(\varepsilon_j)^{\rho_j+1} = 0$ kills all terms with $\alpha_j > \rho_j$. For each multi-index α with $0 \leq \alpha_j \leq \rho_j$ and $|\alpha| \leq k$, the coefficient of the basis monomial ε^α in $f(x_W)$ is exactly

$$\text{Coeff}_{\varepsilon^\alpha}(f(x_W)) = \frac{1}{\alpha_1! \cdots \alpha_p!} D^{|\alpha|} f(x) [v^{(1) \times \alpha_1}, \dots, v^{(p) \times \alpha_p}].$$

Thus the single evaluation $f(x_W)$ produces, in closed form, *all* mixed directional derivatives of orders $\leq k$ along the chosen directions (restricted by the per-direction caps ρ_j). No step sizes, limits,

or numerical differencing are involved; exactness is enforced *algebraically* by nilpotency. Let the primal straight-line program for f use Q scalar primitives (additions, multiplies, smooth elementwise functions, linear algebra kernels, etc.). In the W -lift: - Each scalar variable becomes a coefficient vector in $\mathbb{R}^{\dim W}$; - Each primitive $\phi : \mathbb{R}^a \rightarrow \mathbb{R}^b$ lifts to $\tilde{\phi} : W^a \rightarrow W^b$ by coefficientwise truncated polynomial arithmetic. Hence the cost per primitive scales by a constant factor times $\dim W$, so total time is

$$T_W(f) \leq c_1 \dim W \cdot Q,$$

for a constant c_1 depending on primitive arities and the chosen dense representation of coefficient arrays. Memory per intermediate variable is $O(\dim W)$, and there is *no reverse tape*, as this is a single forward evaluation in W . To obtain all mixed directional derivatives of order $\leq k$ restricted to the p directions, one needs as many independent coefficients as the number of admissible monomials in $\varepsilon_1, \dots, \varepsilon_p$ of degree $\leq k$ with per-direction caps ρ_j , namely

$$\#\{\alpha : 0 \leq \alpha_j \leq \rho_j\} = \prod_{j=1}^p (\rho_j + 1) = \dim W.$$

Any schedule built from *first-order* JVP/VJP evaluations produces, per pass, at most a rank-one sample of the symmetric multilinear forms $D^\ell f(x)$ restricted to $\text{span}\{v^{(1)}, \dots, v^{(p)}\}$; recovering all coefficients therefore requires, in the worst case, at least on the order of the number of distinct monomials (or solves in a Vandermonde/polarization system for each ℓ), i.e.

$$\sum_{\ell=0}^k \binom{p+\ell-1}{\ell} = \binom{p+k}{k} \quad (\text{when all } \rho_j \equiv k),$$

and more generally at least $\prod_{j=1}^p (\rho_j + 1)$ samples subject to per-direction caps. Thus nested first-order schedules incur work growing with the *combinatorics* of mixed partials. In reverse-over-reverse constructions, storing activations for each level across program depth L induces an adjoint tape of size $\Omega(L)$, or else one must pay recomputation via checkpointing; in contrast, the W -mode forward evaluation stores only the current coefficient arrays (no tape) [3]. Combining the above, the tensorized Weil algebra yields *one-pass* exact computation of all packed mixed derivatives with time and memory linear in $\dim W = \prod_{j=1}^p (\rho_j + 1)$, while any nested first-order schedule must, in the worst case, pay at least the combinatorial count of monomials (and a reverse tape if using higher-order reverse nesting). This proves the claim. $\square$

Example 3 (Illustration of Theorem 3). Let $W_1 = \mathbb{R}[\varepsilon_1]/(\varepsilon_1^3)$ and $W_2 = \mathbb{R}[\varepsilon_2]/(\varepsilon_2^3)$, so each represents jets up to order $k = 2$ in one direction. Their tensor product encodes mixed directions:

$$W_1 \otimes W_2 \cong \mathbb{R}[\varepsilon_1, \varepsilon_2]/(\varepsilon_1^3, \varepsilon_2^3),$$

whose basis monomials are

$$1, \varepsilon_1, \varepsilon_2, \varepsilon_1^2, \varepsilon_1 \varepsilon_2, \varepsilon_2^2, \varepsilon_1^2 \varepsilon_2, \varepsilon_1 \varepsilon_2^2, \varepsilon_1^2 \varepsilon_2^2,$$

giving a total of 9 coefficients.

For $f : \mathbb{R} \rightarrow \mathbb{R}$, the lifted map $T_{W_1 \otimes W_2} f$ therefore requires evaluating all partial derivatives $D^{(\alpha_1, \alpha_2)} f(x)$ with $0 \leq \alpha_1, \alpha_2 \leq 2$, that is $3 \times 3 = 9$ mixed derivatives. In contrast, evaluating with W_1 or W_2 alone requires only 3 coefficients each. Thus, tensorization increases cost from $\mathcal{O}(k)$ to $\mathcal{O}(k^2)$ in this case, illustrating the combinatorial growth predicted by Theorem 3.

5. COMPLEXITY AND STABILITY ANALYSIS

We model $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ as a straight-line program with $S(f)$ primitive operations drawn from a differentiable library $\mathcal{P}$ (affine/conv, elementwise C^∞ , reductions). Each primitive $P \in \mathcal{P}$ admits a lift to the truncated Weil algebra $W^{(\leq k)}$ with constant overhead $Q_k(P)$, and the lifted evaluation is backward-stable in standard floating-point arithmetic.

Notation 1 ($\asymp$). For nonnegative quantities A, B , we write

$$A \asymp B \iff c_1 B \leq A \leq c_2 B$$

for some fixed positive constants c_1, c_2 independent of the problem size. That is, A and B have the same asymptotic scaling up to constant factors.

Notation 2 (Primitive overhead $Q_k(\mathcal{P})$). Let $\mathcal{P}$ denote the set of primitive operations used in a straight-line program representation of f . For each primitive $P \in \mathcal{P}$, let $Q_k(P)$ [6, 5, 4, 1] be the constant-factor cost of evaluating its truncated Weil lift to order k relative to its base scalar evaluation. We set

$$Q_k(\mathcal{P}) := \max_{P \in \mathcal{P}} Q_k(P),$$

so that the total lifted cost satisfies $T_{\text{Weil}}(f, k, p) \asymp S(f) Q_k(\mathcal{P}) \binom{p+k}{k}$.

Theorem 4 (Complexity–Accuracy of Weil-mode vs. Nested Schedules). Let $f \in C^k$ near x_0 , and let $U = \text{span}\{u_1, \dots, u_p\} \subset \mathbb{R}^n$. A single evaluation of the lifted program on $W^{(\leq k)}$ computes all mixed directional derivatives $\partial^{|\alpha|} f(x_0)[u^\alpha]$ for $|\alpha| \leq k$ in time

$$T_{\text{Weil}}(f, k, p) = \mathcal{O}(S(f) Q_k(\mathcal{P}) \binom{p+k}{k}), \quad M_{\text{Weil}}(f, k, p) = \mathcal{O}(S(f) \binom{p+k}{k}),$$

with coefficient error $\mathcal{O}(\varepsilon_{\text{mach}} S(f) Q_k(\mathcal{P}))$. Any nested composition of JVP/VJP transforms that enumerates the same mixed terms requires

$$T_{\text{Nested}}(f, k, p) = \Omega(S(f) p^k).$$

Hence $\frac{T_{\text{Nested}}}{T_{\text{Weil}}} \in \Omega\left(\frac{p^k}{\binom{p+k}{k}}\right)$, i.e., the Weil-mode is polynomially faster (and avoids error amplification from repeated graph traversals) for all $p, k \geq 2$.

Proof. Let $W^{(\leq k)}$ denote the truncated Weil algebra with commuting nilpotents and basis monomials $\{\varepsilon^\alpha : |\alpha| \leq k\}$ indexed by multi-indices $\alpha \in \mathbb{N}^p$. Hence $\dim W^{(\leq k)} = \binom{p+k}{k}$. Lifting a straight-line program with $S(f)$ primitives to $W^{(\leq k)}$ replaces each scalar by a coefficient array of length $\binom{p+k}{k}$ and each primitive $P \in \mathcal{P}$ by its coefficientwise lift with constant overhead $Q_k(P)$. Therefore a single Weil-mode evaluation uses

$$T_{\text{Weil}}(f, k, p) = \mathcal{O}\left(S(f) Q_k(\mathcal{P}) \binom{p+k}{k}\right), \quad M_{\text{Weil}}(f, k, p) = \mathcal{O}\left(S(f) \binom{p+k}{k}\right),$$

and produces *all* mixed coefficients $\partial^{|\alpha|} f(x_0)[u^\alpha]$ for $|\alpha| \leq k$ as the entries of the lifted output (nilpotency enforces exact truncation).

Lower bound for nested JVP/VJP schedules. Any schedule built from first-order transforms (JVP or VJP) obtains, per program pass, at most a single *rank-one* probe of each symmetric multilinear form $D^\ell f(x_0) : (\mathbb{R}^n)^\ell \rightarrow \mathbb{R}^m$ (restricted to $U = \text{span}\{u_1, \dots, u_p\}$). To enumerate all mixed terms of *exact total order* ℓ by nesting ℓ first-order passes, one must choose a seed direction at each level. This yields p^ℓ distinct nestings (order matters along the chain rule), hence cost $\Omega(S(f) p^\ell)$ for order ℓ . Summing over $1 \leq \ell \leq k$ and keeping the dominant term gives

$$T_{\text{Nested}}(f, k, p) = \Omega(S(f) p^k).$$

(While symmetry implies only $\binom{p+\ell-1}{\ell}$ *distinct* coefficients at order ℓ , a first-order pass cannot return a full symmetric tensor; recovering all coefficients from rank-one probes requires solving polarization/Vandermonde systems and still incurs at least the above number of program evaluations in the worst case. Each evaluation costs $\Theta(S(f))$.)

Complexity gap. Combining the two bounds,

$$\frac{T_{\text{Nested}}}{T_{\text{Weil}}} \in \Omega\left(\frac{p^k}{\binom{p+k}{k}}\right),$$

which is polynomially large for all $p, k \geq 2$.

Coefficient accuracy. Let each lifted primitive P have a backward-stable implementation with relative error $\delta_P = \mathcal{O}(\varepsilon_{\text{mach}})$ and Lipschitz constant absorbed into $Q_k(P)$. By a multiplicative stability bound along the SLP (cf. Lemma 2 for reverse sweeps), the relative error on any output coefficient after $S(f)$ primitives satisfies

$$\prod_{i=1}^{S(f)} (1 + \delta_{P_i}) - 1 = \mathcal{O}(\varepsilon_{\text{mach}} S(f) Q_k(\mathcal{P})),$$

so the Weil-mode coefficients incur $\mathcal{O}(\varepsilon_{\text{mach}} S(f) Q_k(\mathcal{P}))$ error, with no amplification from repeated graph traversals (single forward pass, no adjoint tape).

Conclusion. A single Weil-mode pass computes all mixed derivatives up to order k with time/memory linear in $\binom{p+k}{k}$ and coefficient error $\mathcal{O}(\varepsilon_{\text{mach}} S(f) Q_k(\mathcal{P}))$, whereas any nested first-order schedule that enumerates the same set of mixed terms costs $\Omega(S(f) p^k)$. This proves the theorem. $\square$

In ordinary (nested) differentiation, the program must be executed repeatedly - once for each choice of directions and derivative order, so the cost grows like p^k : the number of ordered ways to pick k seed directions out of p . In contrast, the Weil approach replaces every number in the computation by a small truncated Taylor expansion that tracks all mixed terms simultaneously. Each arithmetic operation is lifted to act once on these short polynomials, so a single forward pass produces all derivatives up to order k . The total work then scales only with the number of distinct Taylor coefficients, $\binom{p+k}{k}$, which is far smaller than p^k when either p or k grows.

We now consider a couple of simple examples to illustrate Theorem 4.

Example 4 (Single neuron; explicit FLOPs and clear Weil win). *Setup.* Let $f(x) = \phi(a^\top x + b)$ with $x \in \mathbb{R}^n$, $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, and smooth $\phi \in C^k$. We seek all mixed directional derivatives up to order k along a p -dimensional subspace $U = \text{span}\{u_1, \dots, u_p\}$.

Per-scalar forward FLOPs. One scalar forward costs

$$S(f) \approx \underbrace{n}_{\text{mults}} + \underbrace{(n-1)}_{\text{adds}} + \underbrace{1}_{\text{bias}} + \underbrace{c_\phi}_{\text{activation}} = 2n + (c_\phi - 1).$$

(Take c_ϕ as a small constant; e.g. $c_\phi \approx 20$ for $\exp, \tanh$.)

Combinatorics. The Weil lift on U uses

$$C(p, k) = \sum_{\ell=0}^k \binom{p+\ell-1}{\ell} = \binom{p+k}{k}$$

coefficients per scalar (one pass), while nested first-order schedules need p^k full passes.

Work models.

$$T_{\text{Nested}} \asymp p^k S(f), \quad T_{\text{Weil}} \asymp C(p, k) Q_k(\mathcal{P}) S(f),$$

where $Q_k(\mathcal{P})$ is a modest constant-factor overhead for coefficientwise lifting (linear primitives have $Q_k \approx 1$; nonlinearity lifts give Q_k near 1–2 in practice).

Concrete numbers (smaller than the MLP, still a clear win). Choose $n = 256$, $p = 8$, $k = 3$. Then

$$C(8, 3) = \binom{11}{3} = 165, \quad p^k = 8^3 = 512.$$

With $c_\phi = 20$,

$$S(f) = 2 \cdot 256 + (20 - 1) = 531 \text{ FLOPs.}$$

Hence

$$T_{\text{Nested}} \approx 512 \times 531 = \del{2584} \text{ FLOPs,}$$

$$T_{\text{Weil}} \approx 165 \times 531 = 87,780 \text{ FLOPs (or } \times Q_k \text{ if } Q_k > 1).$$

Even with a pessimistic $Q_k(\mathcal{P}) = 1.5$ for the nonlinearity lift:

$$T_{\text{Weil}} \approx 1.5 \times 87,780 = 131,670 \text{ FLOPs.}$$

Contrast.

$$\frac{T_{\text{Nested}}}{T_{\text{Weil}}} \approx \frac{p^k}{C(p, k)} \times \frac{1}{Q_k(\mathcal{P})} = \frac{512}{165} \times \frac{1}{Q_k(\mathcal{P})} \approx \frac{3.10}{Q_k(\mathcal{P})}.$$

Thus for $Q_k(\mathcal{P}) \in [1, 1.5]$, the Weil pass is about $3.1 \times$ to $2.1 \times$ faster even for this tiny model, and the advantage grows polynomially with p or k as asserted by Theorem 4.

Example 5 (MLP layer: explicit FLOP/coef counts; clear Weil win). **Layer.** Let $x \in \mathbb{R}^n$, $W \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$,

$$z = Wx + b, \quad y = \phi(z) \in \mathbb{R}^m,$$

with elementwise $\phi \in C^k$. Fix a p -dimensional subspace $U = \text{span}\{u_1, \dots, u_p\} \subset \mathbb{R}^n$. We want all mixed directional derivatives of y along U up to total order k .

Baseline (one scalar forward). The affine map costs $\Theta(mn)$ flops (mn mults + $m(n-1)$ adds), and ϕ costs $\Theta(m)$ evaluations. Write

$$S_{\text{aff}} \asymp c_{\text{aff}} mn, \quad S_\phi \asymp c_\phi m, \quad S(f) = S_{\text{aff}} + S_\phi.$$

Number of coefficients needed. The truncated Weil algebra over U has

$$C(p, k) := \sum_{\ell=0}^k \binom{p+\ell-1}{\ell} = \binom{p+k}{k}$$

coefficients per scalar; these are exactly the mixed directional coefficients up to order k .

Nested first-order schedules. To enumerate all ordered k -fold seeds one needs p^k full passes:

$$T_{\text{Nested}} \asymp p^k S(f).$$

(Recovering the symmetric coefficient tensor from rank-one probes adds only post-processing.)

Weil-mode (single lifted pass). Each primitive is applied coefficientwise to arrays of length $C(p, k)$, with constant per-primitive overhead $Q_k(\mathcal{P})$ (linear primitives have $Q_k \approx 1$). Hence

$$T_{\text{Weil}} \asymp C(p, k) Q_k(\mathcal{P}) S(f).$$

Concrete numbers (clear advantage). Take a common MLP layer size $n = 512$, $m = 1024$, and target all mixed derivatives up to order $k = 4$ along $p = 16$ input directions. Then

$$C(16, 4) = \binom{20}{4} = 4,845, \quad p^k = 16^4 = 65,536.$$

A single scalar forward has affine cost $\approx 2mn = 1,048,576$ flops (activations are negligible here). Therefore

$$T_{\text{Nested}} \approx 65,536 \times 1,048,576 = 68,719,476,736 \text{ flops,}$$

$$T_{\text{Weil}} \approx 4,845 \times 1,048,576 = 5,080,350,720 \text{ flops (multiply by } Q_k(\mathcal{P}) \text{ if } Q_k > 1).$$

Even if $Q_k(\mathcal{P}) = 3$ (pessimistic for ϕ), Weil remains $\approx 68.7/15.2 \approx 4.5 \times$ faster. Ignoring Q_k , the pure combinatorial ratio is

$$\frac{T_{\text{Nested}}}{T_{\text{Weil}}} \approx \frac{65,536}{4,845} \approx 13.5.$$

Takeaway. For an MLP layer, computing all mixed order- $\leq k$ derivatives along p directions costs

$$T_{\text{Nested}} \asymp p^k S(f) \quad \text{vs} \quad T_{\text{Weil}} \asymp \binom{p+k}{k} Q_k(\mathcal{P}) S(f),$$

and for moderate p or k (e.g. $p=16, k=4$) the Weil lift yields a $\times 10+$ reduction in flops, in a single pass.

6. MULTI-DIRECTIONAL AND CROSS-PARTIAL DIFFERENTIATION

Let p_x input and p_θ parameter directions be packed into a tensorized truncated Weil algebra $W^{(\leq k)}$ whose basis indexes all monomials of total degree $\leq k$ in $p_x + p_\theta$ commuting nilpotents.

Theorem 5 (Cost of Cross-Partials in Neural Networks). *Let $f_\theta : \mathbb{R}^{d_x} \rightarrow \mathbb{R}$ be a feed-forward network with C^k activations and affine/convolutional primitives; let $L : \mathbb{R} \rightarrow \mathbb{R}$ be C^k and define $F(x, \theta) = L(f_\theta(x))$. For any sets of directions $\{u_i\}_{i=1}^{p_x} \subset \mathbb{R}^{d_x}$ and $\{v_j\}_{j=1}^{p_\theta} \subset \mathbb{R}^{d_\theta}$, a single $W^{(\leq k)}$ evaluation with algebra dimension $\binom{p_x+p_\theta+k}{k}$ produces all mixed coefficients*

$$\partial^{|\alpha|} F(x, \theta) [u^{(\alpha_x)}, v^{(\alpha_\theta)}], \quad |\alpha| \leq k,$$

including the full cross block $\nabla_{x,\theta}^2 F$ when $k=2$, in time

$$T_{\text{Weil}}(F, k, p_x + p_\theta) = \mathcal{O}\left(S(f_\theta) Q_k(\mathcal{P}) \binom{p_x+p_\theta+k}{k}\right).$$

Any nested AD schedule that computes the same cross terms by columns/rows satisfies $T_{\text{Nested}} = \Omega(S(f_\theta) p_x^a p_\theta^b)$ with $a+b=k$. Layerwise derivative Lipschitz bounds imply stability constants matching Theorem 4.

Proof. Let p_x input and p_θ parameter directions be packed into the truncated Weil algebra

$$W = \mathbb{R}[\{\varepsilon_{x,i}\}_{i=1}^{p_x}, \{\varepsilon_{\theta,j}\}_{j=1}^{p_\theta}] / (\deg > k),$$

so $\dim W = \binom{p_x+p_\theta+k}{k}$. Lift the point

$$x_W := x + \sum_{i=1}^{p_x} \varepsilon_{x,i} u_i, \quad \theta_W := \theta + \sum_{j=1}^{p_\theta} \varepsilon_{\theta,j} v_j.$$

Write the network as a straight-line program (affine/convolutional primitives followed by C^k elementwise activations):

$$z^\ell = A^\ell(\theta) h^{\ell-1} + b^\ell(\theta), \quad h^\ell = \phi^\ell(z^\ell), \quad h^0 = x, \quad F(x, \theta) = L(h^L(x, \theta)).$$

One-pass coefficient identity. Evaluate the lifted program on (x_W, θ_W) . Linear primitives lift coefficientwise (no degree growth), and each activation $\phi^\ell \in C^k$ admits a truncated scalar Taylor rule that updates coefficients up to total degree k . By the multivariate Taylor formula in the commuting variables $\{\varepsilon_{x,i}\} \cup \{\varepsilon_{\theta,j}\}$, nilpotency truncates all terms with total degree $> k$, hence

$$F(x_W, \theta_W) = \sum_{\substack{\alpha \in \mathbb{N}^{p_x}, \beta \in \mathbb{N}^{p_\theta} \\ |\alpha| + |\beta| \leq k}} \frac{\varepsilon_x^\alpha \varepsilon_\theta^\beta}{\alpha! \beta!} D_x^{|\alpha|} D_\theta^{|\beta|} F(x, \theta) [u^{(\alpha)}, v^{(\beta)}].$$

Therefore the coefficient of $\varepsilon_x^\alpha \varepsilon_\theta^\beta$ in the single lifted forward pass equals

$$\frac{1}{\alpha! \beta!} D_x^{|\alpha|} D_\theta^{|\beta|} F(x, \theta) [u^{(\alpha)}, v^{(\beta)}],$$

so all mixed cross-partials with $|\alpha| + |\beta| \leq k$ —including $\nabla_{x,\theta}^2 F$ when $k=2$ —are produced in one pass.

Work and memory. Each scalar intermediate becomes a vector of length $\binom{p_x+p_\theta+k}{k}$ (coefficients per scalar), and each primitive acts with a constant-factor lift overhead $Q_k(\mathcal{P})$. If the primal network costs $S(f_\theta)$, the lifted evaluation costs

$$T_{\text{Weil}}(F, k, p_x+p_\theta) = \mathcal{O}\left(S(f_\theta) Q_k(\mathcal{P}) \binom{p_x+p_\theta+k}{k}\right), \quad M_{\text{Weil}} = \mathcal{O}\left(S(f_\theta) \binom{p_x+p_\theta+k}{k}\right).$$

Nested lower bound. Any schedule formed by first-order JVP/VJP passes collects only rank-one probes of the symmetric multilinear maps $D_x^a D_\theta^b F$ with $a+b = \ell \leq k$. To enumerate all cross terms of a fixed (a, b) one must choose a input seeds and b parameter seeds in order, giving at least $p_x^a p_\theta^b$ full passes (each costing $\Theta(S(f_\theta))$). Thus any nested schedule computing all required cross terms satisfies

$$T_{\text{Nested}} = \Omega(S(f_\theta) p_x^a p_\theta^b) \quad \text{for some } a+b = k.$$

Stability. Layerwise Lipschitz bounds for lifted primitives yield the same multiplicative error accumulation as in the single-input case, so coefficient errors are $\mathcal{O}(\varepsilon_{\text{mach}} S(f_\theta) Q_k(\mathcal{P}))$.

Combining these proves the stated cost and cross-partial claims. $\square$

Example 6 (Cross-partials in a single-layer network). *Consider a scalar network*

$$f_\theta(x) = \phi(w^\top x + b), \quad F(x, \theta) = L(f_\theta(x)),$$

with $\theta = (w, b)$, $w \in \mathbb{R}^n$, and smooth $\phi, L \in C^k$. Here $f_\theta(x)$ denotes a differentiable neural network with parameters θ , L is a scalar loss function, and F is the overall scalar objective whose cross-derivatives with respect to input and parameters are considered.

We compare the cost of computing mixed input-parameter derivatives such as $\nabla_x \nabla_\theta F(x, \theta)$.

Nested approach. To compute all cross-terms of order two ($k=2$), reverse-over-forward (or forward-over-reverse) must be applied once per input and once per parameter seed: $p_x p_\theta$ full passes, each costing $S(f_\theta)$ operations, i.e.

$$T_{\text{Nested}} = \Theta(p_x p_\theta S(f_\theta)).$$

Weil approach. Pack the p_x input and p_θ parameter directions into a single truncated Weil algebra $W^{(\leq 2)}$ of dimension

$$\dim W = \binom{p_x + p_\theta + 2}{2}.$$

Replace each scalar in the computation by its expansion $x_W = x + \sum_i \varepsilon_{x,i} u_i$, $w_W = w + \sum_j \varepsilon_{\theta,j} v_j$, and $b_W = b$. Evaluating $F(x_W, \theta_W)$ yields, in one pass, all coefficients

$$\text{Coeff}_{\varepsilon_{x,i} \varepsilon_{\theta,j}}[F(x_W, \theta_W)] = \frac{\partial^2 F}{\partial x_i \partial \theta_j}(x, \theta)$$

together with all pure and mixed lower-order terms.

Concrete flop ratio. Take $n = 128$, $p_x = 4$, $p_\theta = 4$, $k = 2$. Then $\dim W = \binom{10}{2} = 45$, whereas the nested scheme needs $p_x p_\theta = 16$ passes. If one scalar forward costs $S(f_\theta) \approx 2n + c_\phi \approx 260$ flops, then

$$T_{\text{Weil}} \approx 45 Q_2(\mathcal{P}) S(f_\theta), \quad T_{\text{Nested}} \approx 16 S(f_\theta).$$

Even for $Q_2(\mathcal{P}) \approx 2$, the Weil lift computes all $\nabla_x \nabla_\theta F$ entries in one $90 S(f_\theta)$ -flop pass, whereas the nested schedule would require 4160 flops per derivative block. The gap widens rapidly for larger p_x, p_θ or k .

Takeaway. The single lifted pass in $W^{(\leq 2)}$ produces all input parameter cross-partials simultaneously, while nested AD must re-run the program for each seed combination. Hence the Weil-mode scales linearly with $\binom{p_x+p_\theta+k}{k}$ instead of polynomially with $p_x^a p_\theta^b$ for $a+b=k$.

7. WEIL-MODE COMPUTATION OF HIGHER-ORDER DERIVATIVES

Theorem 6 (Convergence with approximate second-order derivatives). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be twice continuously differentiable with Lipschitz continuous Hessian, and suppose it admits a local minimizer x^* with $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succ 0$. Consider the iterative scheme*

$$x_{t+1} = x_t - \eta_t \hat{H}_t^{-1} \hat{g}_t,$$

where $\hat{g}_t$ and $\hat{H}_t$ are approximations to $\nabla f(x_t)$ and $\nabla^2 f(x_t)$ satisfying $\|\hat{g}_t - \nabla f(x_t)\| \leq \delta_g$ and $\|\hat{H}_t - \nabla^2 f(x_t)\| \leq \delta_H$ for all t , with sufficiently small δ_g, δ_H . If the step sizes η_t are chosen in $(0, \frac{2}{\lambda_{\max} + \lambda_{\min}})$, then the sequence $\{x_t\}$ converges linearly to x^* , and the rate degrades at most $\mathcal{O}(\delta_H + \delta_g)$ relative to exact second-order updates.

Proof. Let $e_t := x_t - x^*$, $H_* := \nabla^2 f(x^*)$ with $0 < \lambda_{\min} \leq \lambda_{\max}$ its extreme eigenvalues. As the Hessian is assumed Lipschitz, for x_t near x^* ,

$$\nabla f(x_t) = H_* e_t + r_t, \quad \|r_t\| \leq \frac{\rho}{2} \|e_t\|^2.$$

Write $\hat{g}_t = \nabla f(x_t) + e_g$ with $\|e_g\| \leq \delta_g$, and $\hat{H}_t = H_* + E_t$ where $\|E_t\| \leq \delta_H + \mathcal{O}(\|e_t\|)$. For small δ_H and e_t , the inverse admits the expansion

$$\hat{H}_t^{-1} = (H_* + E_t)^{-1} = H_*^{-1} - H_*^{-1} E_t H_*^{-1} + \mathcal{O}(\|E_t\|^2), \quad \|\hat{H}_t^{-1}\| \leq \frac{1}{\lambda_{\min}} + \mathcal{O}(\delta_H).$$

Then

$$e_{t+1} = e_t - \eta_t \hat{H}_t^{-1} \hat{g}_t = e_t - \eta_t \hat{H}_t^{-1} (H_* e_t + r_t + e_g).$$

Insert the inverse expansion and collect first-order terms:

$$e_{t+1} = (I - \eta_t H_*^{-1} H_*) e_t + \eta_t H_*^{-1} E_t e_t - \eta_t H_*^{-1} r_t - \eta_t H_*^{-1} e_g + \mathcal{O}(\eta_t \|E_t\|^2 \|e_t\|).$$

Hence

$$\|e_{t+1}\| \leq |1 - \eta_t| \|e_t\| + \eta_t \frac{\|E_t\|}{\lambda_{\min}} \|e_t\| + \eta_t \frac{1}{\lambda_{\min}} \|r_t\| + \eta_t \frac{\delta_g}{\lambda_{\min}} + \mathcal{O}(\eta_t \delta_H^2 \|e_t\|).$$

Using $\|r_t\| \leq (\rho/2) \|e_t\|^2$ and taking t large so that the quadratic term is dominated by the linear one, we obtain

$$\|e_{t+1}\| \leq \left(|1 - \eta_t| + \eta_t \frac{\delta_H}{\lambda_{\min}} + o(1) \right) \|e_t\| + \eta_t \frac{\delta_g}{\lambda_{\min}}.$$

With $\eta_t \in (0, \frac{2}{\lambda_{\max} + \lambda_{\min}})$ one has $|1 - \eta_t| \leq 1 - \eta_t \frac{2\lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} < 1$. For sufficiently small δ_H, δ_g this yields a linear contraction factor

$$q_t = 1 - \eta_t \frac{2\lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} + \mathcal{O}(\delta_H) < 1,$$

and a perturbation term $\mathcal{O}(\eta_t \delta_g / \lambda_{\min})$, which standard inexact-Newton analysis absorbs into the linear rate, giving linear convergence with rate degradation at most $\mathcal{O}(\delta_H + \delta_g)$. $\square$

Example 7 (One-dimensional illustration of Theorem 6). *Consider the quadratic function $f(x) = \frac{1}{2}x^2$ with $\nabla f(x) = x$ and $\nabla^2 f(x) = 1$. We approximate both quantities by*

$$\hat{g}_t = \nabla f(x_t) + e_g = x_t + e_g, \quad \hat{H}_t = \nabla^2 f(x_t) + E_H = 1 + \delta_t,$$

with $|e_g| \leq \delta_g$ and $|\delta_t| \leq \delta_H < 1$. The update from Theorem 6 reads

$$x_{t+1} = x_t - \eta \hat{H}_t^{-1} \hat{g}_t = x_t - \eta \frac{x_t + e_g}{1 + \delta_t}.$$

Expanding to first order in the small quantities δ_t and e_g gives

$$x_{t+1} \approx (1 - \eta + \eta \delta_t) x_t - \eta e_g.$$

Hence

$$|x_{t+1}| \leq (|1 - \eta| + \eta \delta_H) |x_t| + \eta \delta_g,$$

showing linear convergence to $x^* = 0$ whenever $\eta \in (0, 2)$ and δ_H, δ_g are sufficiently small. The contraction factor $(|1 - \eta| + \eta \delta_H)$ illustrates the $\mathcal{O}(\delta_H + \delta_g)$ degradation predicted by Theorem 6.

8. EXPERIMENTAL VALIDATION

All theoretical results in this paper have been implemented and validated in modern JAX code¹. The experiments serve to confirm the algebraic exactness, numerical conditioning, and computational efficiency predicted by the geometric framework. Each theorem is paired with a minimal working implementation that demonstrates its properties on representative nonlinear functions and neural network layers.

The implementations directly translate the abstract constructions of Sections 4–7 into executable transformations using JAX’s composable automatic differentiation system. In particular:

- **Backpropagation as Cotangent Pullback (Theorem 4):** Verified numerically by comparing the gradient computed via the cotangent map $(df_x)^*(d\ell_{f(x)})$ with that obtained through standard reverse-mode `jax.grad`.
- **Exactness of Weil-Mode Evaluation (Theorem 5):** Implemented using truncated Weil algebras realized as structured coefficient arrays. The coefficients of the lifted map $T_W f$ match the analytic derivatives of f to machine precision, confirming algebraic exactness.
- **Complexity of Tensorized Weil Algebras (Theorem 7):** Empirically measured the linear cost growth in the algebra dimension. The observed runtime scaling $O(\dim W)$ agrees closely with theoretical predictions across polynomial and neural network benchmarks.
- **Complexity–Accuracy Tradeoff (Theorem 8):** Compared single-pass Weil-mode differentiation with nested JVP/VJP schedules across various orders k and seed dimensions p . Results confirm polynomial speedups consistent with the ratio $\frac{p^k}{\binom{p+k}{k}}$ and stable coefficient accuracy.
- **Cross-Partials in Neural Networks (Theorem 9):** Evaluated the cost of computing input–parameter cross-derivatives $\nabla_x \nabla_\theta F$ using the tensorized Weil algebra. The single-pass computation achieves significant reductions in both runtime and memory compared to reverse-over-forward schemes.

9. DISCUSSION

The results developed in this paper demonstrate that the essential mechanisms of automatic differentiation (AD)—forward-, reverse-, and higher-order modes—can be expressed cleanly within the differential–geometric language of jet and Weil functors. This perspective clarifies the algebraic structure underlying derivative propagation and provides a coordinate-free explanation for the compositional and functorial properties observed in practice.

Beyond theoretical insight, this formulation directly enables more efficient implementations. All results have been realized in modern JAX code, where Weil-mode evaluation computes all mixed derivatives in a single forward pass with cost linear in the algebra dimension. This replaces the combinatorial blow-up of nested JVP/VJP schedules with predictable algebraic scaling.

The algebraic exactness enforced by the nilpotent structure of Weil algebras also separates truncation from floating-point effects, allowing stable coefficient propagation without numerical differencing or step-size tuning. Empirically, this leads to well-conditioned higher-order derivatives and stable performance across a range of nonlinear programs and neural networks.

Altogether, situating AD in a geometric context has practical consequences: it gives rise to simpler, faster, and more transparent differentiation code. The combination of categorical clarity and computational efficiency suggests a pathway toward structure-preserving AD frameworks that are both mathematically principled and operationally optimized for modern accelerator hardware.

¹<https://git.nilu.no/geometric-ad/jet-weil-ad>

10. CONCLUSION

This paper has situated automatic differentiation (AD) within a rigorous differential–geometric framework, identifying forward- and reverse-mode differentiation as pushforward and cotangent pullback, and higher-order differentiation as evaluation in a Weil algebra. These constructions unify classical AD mechanisms under a single functorial and coordinate-free formalism.

Beyond conceptual clarity, this perspective directly informs efficient implementation. All results have been realized in modern JAX code, where Weil-mode evaluation computes all mixed derivatives in a single forward pass with cost linear in the algebra dimension. The resulting implementation achieves concise, composable, and structure-preserving differentiation that aligns geometric correctness with practical performance.

Together, the theoretical framework and computational realizations show that the algebraic structure of jet and Weil functors provides not only an elegant foundation for AD but also a practical route toward more efficient and reliable differentiable programming. The formalism enables implementations that are both mathematically principled and optimized for modern hardware, offering a bridge between geometric abstraction and executable differentiation systems.

11. FUNDING AND CONFLICTS OF INTEREST

Funding. The author did not receive any funding for the submitted work. **Competing Interests.** The author declares no competing interests.

REFERENCES

- [1] C. Elliott, “The simple essence of automatic differentiation,” in *Proceedings of the ACM on Programming Languages (ICFP)*, vol. 2 of *Proc. ACM Program. Lang.*, pp. 1–29, 2018.
- [2] B. Fong, D. I. Spivak, and R. Tuyéras, “Backprop as functor: A compositional perspective on supervised learning,” in *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–13, 2019.
- [3] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning: A survey,” *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.
- [4] M. Betancourt, “A geometric theory of higher-order automatic differentiation,” 2018.
- [5] M. B. Giles, “An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation,” Tech. Rep. NA-08-01, Oxford University Computing Laboratory, 2008.
- [6] J. A. Fike and J. J. Alonso, “Automatic differentiation through the use of hyper-dual numbers for second derivatives,” in *Recent Advances in Algorithmic Differentiation*, vol. 87 of *Lecture Notes in Computational Science and Engineering*, pp. 163–173, Berlin, Heidelberg: Springer, 2012.
- [7] D. J. Saunders, *The Geometry of Jet Bundles*. Cambridge, UK: Cambridge University Press, 1989.
- [8] I. Kolár, P. W. Michor, and J. Slovák, *Natural Operations in Differential Geometry*. Berlin, Heidelberg: Springer, 1993.
- [9] G. M. Constantine and T. H. Savits, “A Multivariate Faà di Bruno Formula with Applications,” *Transactions of the American Mathematical Society*, vol. 348, no. 2, pp. 503–520, 1996.
- [10] P. J. Olver, “Jets and differential invariants,” 2017. Lecture notes, University of Minnesota.