Research manuscript

Cotangent Semantics for Distributed Optimization

Schedule coherence and reproducible benchmarks on three QEMU guests

Anonymous submission

Abstract. A distributed smooth program can expose its reverse dependency graph as a protocol for cotangent messages. We construct a semantics in which legal schedules of a typed acyclic circuit are related by coherent 2-cells. A symmetric monoidal reverse interpretation sends every legal schedule to the same cotangent lens, with transposed derivatives and additive cotangent fan-in. A finite-fiber base-change identity describes aggregation under reindexing, and the interpretation descends for compatible smooth families over a context site. We derive synchronous minibatch SGD from the distributed pullback and implement the forward and reverse traffic on three QEMU guests. At 18,432 training example evaluations per configuration, SGD gave the lowest mean test log loss on Iris with radial basis features, handwritten digits with radial basis features and with a tanh network, and pooled Fashion-MNIST. The comparison includes a real-coded genetic algorithm, an entropy-maximizing additive boosting rule, and uniform and frozen-feature controls. Two independently booted cluster runs reproduced all logged losses, accuracy counts, derivative checks, and application-byte counts exactly.

Keywords. Reverse-mode differentiation, categorical semantics, distributed optimization, stochastic gradient descent, entropy regularization, reproducibility

1 Introduction

A distributed implementation exposes more structure than its extensional input–output map. Independent operations can finish in different orders. A feature shard can be moved to a different machine. A parameter can contribute to several outputs, so several messages must be combined before a reverse operation can execute. The semantics has to say which changes preserve the computed differential and which affect the cost of computing it. We address this question for finite smooth programs whose primal trace and vector–Jacobian products are available.

Reverse derivative categories provide an axiomatic setting for reverse differentiation [1]. Parametric lenses describe how learning rules act on models [2]. Adjoinable MPI studies the reversal of communication in numerical programs [3]. Our construction relates these ideas at the level of typed schedules. It gives a two-dimensional presentation of reorderings, a cotangent correctness theorem with explicit wire occurrences, and a base-change law for finite aggregation. The presentation also specifies what is preserved when a forward and reverse exchange is placed across a virtual network.

The experiments ask two separate questions. First, do actual cross-guest cotangents agree with independent finite differences and survive a change in scheduling? Second, what happens when full SGD, a forward-only genetic algorithm, and a boosting rule using entropy-maximizing example weights are given the same number of forward evaluations? The boosting rule adds to a finite dictionary of weak learners. Its weighting policy has a precise variational characterization, and a uniform-weight ablation isolates that policy.

2 Typed circuits and their dual

2.1 Smooth primitives and occurrences

An interface is an ordered list of finite-dimensional real vector spaces, interpreted as their product. Execution uses fixed coordinates on each factor. The empty list is interpreted by the zero-dimensional space. An open circuit has a finite acyclic event graph with ordered input and output ports. Each event v carries a smooth map f_v from the product of its input types to the product of its output types. Each input port has exactly one source wire. An output wire may be unused or may feed multiple ports, including repeated inputs to one event. Numbering consumers by *port occurrence* retains that multiplicity. Boundary output ports count as consumers. We write $F_C \colon X \to Y$ for the resulting smooth map of a circuit C.

Parameter and data inputs may be written $X = P \times A$. Shared parameters are wires with multiple consumers. Batching is a finite product of per-example computations with copies of a shared parameter input and a final loss aggregation. The smoothness requirement applies to parameters for each fixed data record and each fixed branch of the training protocol. We use C^1 for the reverse semantics and additional smoothness assumptions for optimization estimates. Discrete labels and minibatch indices are held fixed during differentiation.

For a smooth map $f \colon X \to Y$, its reverse at x is the linear map

$$f_x^* \colon Y^* \longrightarrow X^*, \qquad f_x^* \bar{y} = \bar{y} \circ Df_x.$$

A reverse lens from X to Y consists of a smooth forward map f and a smoothly varying, fiberwise linear backward map $r_x \colon Y^* \to X^*$. The composite of (f, r) followed by

(g, s) is

$$(g, s) \circ (f, r) = (g \circ f,\ x \mapsto r_x\, s_{f(x)}).$$

We use smooth maps in the lens category and the corresponding C^1, continuous-backward variant when only first differentiability is assumed. The product of lenses uses the product forward map and the direct sum of backward maps. This defines CotLens, with reverse differentiation $f \mapsto (f, f^*)$ respecting composition and products by the chain rule. Standard coordinates identify a cotangent with its Euclidean gradient when an optimizer needs a vector.

Definition 2.1 (Dual graph)**.** The dual of an executed circuit has a reverse edge for each primal port occurrence. Its value is the cotangent sent from that consumer to the source. The reverse event uses the primal inputs, or equivalent saved data, to apply f_v^*. At every wire, the event waits for its required cotangents and sums them. A boundary seed contributes once per output occurrence. A wire with no contribution receives zero.

For example, if a scalar input is used in both arguments of multiplication, the derivative of $m \circ \Delta$ at x, with $m(a, b) = ab$, is $2x$. In the dual graph, each argument sends $x\bar{y}$ to the same source. A graph that deduplicates those messages returns $x\bar{y}$. If x is duplicated onto n destinations, $\Delta(x) = (x, \dots, x)$ has pullback $(\bar{x}_i)_i \mapsto \sum_i \bar{x}_i$. Conversely the pullback of an addition node sends its output cotangent to every input. Identity transport and wire permutations pull back by the transpose of their routing maps. The operation does not require the forward primitive to be invertible.

2.2 Correctness and communication

Theorem 2.2 (Distributed pullback)**.** *Fix a circuit C, input x, and output seed $\bar{y}$. Any legal forward evaluation followed by a reverse evaluation that delivers each contribution exactly once returns $DF_C(x)^*\bar{y}$. The result is invariant under event placement provided communication implements the specified wire maps.*

Proof. For any input variation $\dot{x}$, propagate a tangent $\dot{w}$ along every wire by applying Df_v to the tuple of event-input tangents. Pair the resulting boundary output tangents with $\bar{y}$. During reverse evaluation, replace each pairing of event-output cotangents with event-output tangents by the corresponding pairing of f_v^* with event-input tangents. This follows from the definition of the transpose. At each wire collect all pairings with its common tangent by additivity. Reverse acyclicity eventually leaves only the boundary input pairings. The sum of those pairings equals $\langle \bar{y}, DF_C(x)\dot{x} \rangle$ for every $\dot{x}$. Nondegeneracy of the pairing gives the claim. A transmission of the same typed value changes none of the pairings. For a nonidentity wire map its transpose enters the same argument. □

This proof models a complete logical trace. Ordered TCP gives the byte stream used by our implementation. Its requests also name a sequence, task, row count, and the forward tape identifier, which is consumed once by a reverse or statistics operation. A lost or stale tape invalidates that iteration. In real arithmetic the sum is independent of its order. The implementation fixes operand order at the coordinator so that changing the worker dispatch order does not change the floating-point sum. An asynchronous system with duplicate retries would need occurrence-level deduplication or a protocol establishing exactly-once contributions.

3 Schedules and higher coherence

3.1 A locally groupoidal 2-category

Let Circ be the category with typed interfaces as objects and open circuits up to boundary-preserving isomorphism as arrows. An isomorphism preserves the primitive labels, port orders, and wiring. Composition glues matching boundary ports, and its isomorphism class is independent of the order of gluing. The tensor is disjoint union of events and concatenation of interfaces, with the symmetry induced by reordering boundary wires. A placement may be carried as a vertex label when the schedule is to be compared at fixed deployment. Erasing that label is a functor.

A schedule is a linear extension of the precedence order on events. Let Sched have the same objects and scheduled circuits, up to schedule-preserving isomorphism, as 1-cells. For any two 1-cells with the same underlying circuit in Circ, give the hom-category exactly one invertible 2-cell between them. It represents the identification of legal reschedulings. For distinct underlying circuits there are no such 2-cells. Concatenation after gluing composes 1-cells. Vertical composition of 2-cells is forced by uniqueness. Horizontal composition is well defined since equality in Circ is a congruence for gluing. Interchange, associativity and the unit axioms follow from this description.

Lemma 3.1 (Adjacent exchanges)**.** *Every rescheduling 2-cell is generated by exchanges of adjacent incomparable events.*

Proof. Fix a target linear extension. In the current extension move its first event to the first position by adjacent swaps. Each event passed is incomparable with it. If a passed event preceded it in the order, the target could not start with the chosen event. If it followed it, the current extension could not place it earlier. Remove the common first event and repeat on the remaining finite order. □

Equivalently, take the groupoid of paths of such exchanges and identify parallel paths. This quotient records schedule equivalence and omits the history of swaps. Tensoring 1-cells concatenates their independent schedules. Its compositor identifies two legal schedules on the same glued disjoint circuit. The unit and symmetry comparisons use the empty schedule and boundary

permutation. The coherence diagrams commute since their comparisons have the same boundary permutation and there is at most one 2-cell between their scheduled circuits.

Theorem 3.2 (Coherent reverse interpretation)**.** *Reverse differentiation induces a symmetric monoidal interpretation of* Sched *in the locally discrete 2-category of cotangent lenses. It factors through* Circ*, and forgetting schedules induces an equivalence on the homotopy-coherent nerves of the corresponding categories enriched in spaces.*

Proof. For each scheduled circuit C, assign the lens (F_C, DF_C^*). The extensional forward evaluation depends only on wiring. Theorem 2.2 gives equality of reverse evaluations for any two schedules of that wiring. Gluing is composition of the forward maps, and its reverse is the composite of the transposes in reverse order. Disjoint union gives the product lens, so the tensor and its comparison 2-cells are preserved. The hom-groupoid of Sched maps to the discrete hom-set of Circ. Each fiber is a nonempty groupoid with exactly one arrow between any two objects, hence has contractible nerve. This gives a weak equivalence on every simplicial mapping space and identity on objects. The homotopy-coherent nerve gives the stated equivalence [4, Sec. 1.1.5]. □

This is a particular higher semantics with contractible spaces of schedule choices. We quotient reorderings of a fixed dependency graph. Graph rewrites, timings, partial failure histories and resource allocations can give finer observations. Theorem 3.2 only identifies the value and cotangent semantics. Our executable audit compares two worker dispatch orders on identical inputs and stores cost measurements independently.

3.2 Indexed aggregation and descent

Let R be a commutative ring and let $u : I \to J$ be a finite map of indexing sets. On families of R-modules, reindexing u^* has a left adjoint Σ_u given by fiberwise direct sums and a right adjoint Π_u given by fiberwise products. The canonical map from sum to product is an isomorphism for finite fibers, including an empty fiber. For a repeated module M_j, the induced replication map has diagonal components $M_j \to \prod_{u(i)=j} M_j$. When these modules are finite free, the dual map has summation components $\bigoplus_{u(i)=j} M_j^* \to M_j^*$.

Proposition 3.3 (Base change for reverse aggregation)**.** *Given a pullback square of finite sets*

$$\begin{array}{ccc} I' & \xrightarrow{a} & I \\ {\scriptstyle u'}\downarrow & & \downarrow{\scriptstyle u} \\ J' & \xrightarrow[b]{} & J \end{array}$$

the comparison $b^*\Sigma_u \cong \Sigma_{u'}a^*$ identifies the two reverse aggregations under fiber reindexing. It is compatible with pasting pullback squares.

Proof. At $j' \in J'$, a induces a bijection $(u')^{-1}(j') \to u^{-1}(b(j'))$. Reindexing the summands by this bijection gives the comparison. Both folds add exactly those summands, with the empty fold zero. Composing pullback squares composes their fiber bijections. The argument works for weighted linear maps when weights are transported with their occurrences. □

For context variation, take the site of opens of a smooth manifold B. Let each interface over an open U be a smooth vector bundle, with products taken over U. A primitive is a smooth bundle map over U that is allowed to be nonlinear on the fibers. Let $T_v f$ denote its vertical differential and $T_v f^*$ the pointwise dual of that differential. Both commute with restriction to smaller opens. In overlapping coordinates the usual chain rule transports $T_v f$ and the dual transition maps transport the cotangents. This construction is interpreted in sheaves over B, or in their embedding as 0-truncated objects of $\mathrm{Shv}(B; \mathcal{S})$. It gives a topos-theoretic account of compatible local programs, with no derivative assigned to an arbitrary internal morphism.

Proposition 3.4 (Descent in the parameter context)**.** *Fix the finite circuit and its port occurrences over B. Compatible smooth local primitive maps on an open cover, with their vector bundle transition functions, define a global forward map and a global vertical cotangent pullback. The global pullback restricts to the composite local dual protocol.*

Proof. Vector bundle sections and smooth maps over B glue uniquely when the local maps agree under the given transitions. In particular the local forward interpretations glue. The vertical chain rule makes the local dual maps agree on each overlap. Finite sums and composition of smooth maps commute with restriction, so the local execution of the dual also glues. On each member of the cover it is the vertical cotangent pullback by Theorem 2.2, applied fiberwise. Uniqueness of the glued map identifies the two global maps. □

The finite-fiber law concerns how indexed contributions are regrouped. For example, partitioning a batch into unequal blocks requires multiplying each local mean by its block size before taking the global mean. An unweighted mean of local means changes the objective whenever block sizes differ and the block means are unequal. The experiments instead shard the model and features, and their output reduction is a sum of partial logits.

4 Optimization from the dual

Let $f_\theta(x) \in \mathbb{R}^K$ be the logits and $\ell_i(\theta) = \mathrm{lse}(f_\theta(x_i)) - f_\theta(x_i)_{y_i}$, where $\mathrm{lse}(z) = \log \sum_c e^{z_c}$. For n training examples, set

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell_i(\theta) + \frac{\lambda}{2} \|\theta\|^2.$$

A batch $I = (i_1, \dots, i_B)$ is sampled independently with replacement from the training indices. The pullback of its mean loss uses seed 1 and returns

$$g_I(\theta) = \frac{1}{B} \sum_{a=1}^{B} \nabla \ell_{i_a}(\theta) + \lambda\theta.$$

Each worker computes its parameter block at the parameter version from the forward pass. For shared parameter blocks, the result includes the transpose of the replication map. The coordinator then applies $\theta \leftarrow \theta - \eta g_I(\theta)$. The Euclidean update depends on the chosen metric, even though the cotangent pullback is natural under smooth coordinate changes. A change of coordinates must carry the metric along to compare intrinsic gradient directions.

Proposition 4.1 (Synchronous SGD)**.** *For an independent uniform minibatch and fixed θ, $\mathbb{E}[g_I \mid \theta] = \nabla\mathcal{L}(\theta)$. If $\mathcal{L}$ is L-smooth along the update segments and the trace of the single-example gradient covariance at the visited θ is at most σ^2, then for $0 < \eta \le 1/L$,*

$$\mathbb{E}[\mathcal{L}(\theta - \eta g_I) \mid \theta] \le \mathcal{L}(\theta) - \frac{\eta}{2}\|\nabla\mathcal{L}(\theta)\|^2 + \frac{L\eta^2\sigma^2}{2B}.$$

Proof. Linearity of expectation gives unbiasedness, and independence gives conditional variance at most σ^2/B. Substitute $v = -\eta g_I$ in $\mathcal{L}(\theta + v) \le \mathcal{L}(\theta) + \langle \nabla\mathcal{L}(\theta), v\rangle + L\|v\|^2/2$. Taking the conditional expectation and using $L\eta \le 1$ gives the claim. □

The bound uses the stipulated smoothness and step size, as in standard stochastic optimization [5]. The benchmark chooses the learning rate on validation loss. For a linear softmax head on fixed features, the Hessian is positive semidefinite before regularization. With ridge on all parameters it is at least λI, and its upper bound is finite on a finite feature matrix. A bound $L \le \lambda + \max_i \|\tilde{x}_i\|^2$ follows by using the bias-augmented feature vector $\tilde{x}_i$ and the inequality $0 \preceq \operatorname{diag}(p) - pp^\top \preceq I$. The tanh network is nonconvex. Its empirical comparison does not require a global smoothness or global convergence claim.

The optimizer acts on the state after evaluating a circuit and its cotangent. Write S for a standard Borel space of parameters and optimizer state, and Ω for the space of draws. For a schedule α, let $U_\alpha(s, \omega)$ apply the deterministic update given the objective value and cotangent, with the batch and any other randomness fixed by ω. The resulting state evolution is the Markov kernel

$$K_\alpha(s, A) = \int_\Omega \mathbf{1}_A\big(U_\alpha(s, \omega)\big)\, \mu_s(d\omega),$$

where μ_s is the stipulated sampling kernel. The genetic population and mutation draws, and the boosting dictionary index, define analogous state transitions [6]. Hard rank selection, tie breaking, and accuracy evaluation are measurable operations. Cotangents differentiate the smooth objective conditional on its inputs.

Proposition 4.2 (Schedule invariance of the update law)**.** *For fixed sampling and an update that depends only on the optimizer state, objective and cotangent, two schedules identified in* Sched *yield the same one-step Markov kernel. The joint law of any finite sequence of such updates is also the same when each schedule may be chosen anew at each step.*

Proof. For each fixed draw, the forward and backward interpretations agree by Theorem 3.2. The state updates are consequently equal pointwise, and integrating against the same sampling kernel gives equal transition probabilities. Coupling the iterates by the same draw conditional on the current state makes their states agree inductively. The same reasoning allows the schedule to depend on earlier draws and observed states as long as it preserves the circuit and the conditional sampling kernel. □

This result concerns the law of the model state. When a scheduler changes the sample stream, accepts stale snapshots, or changes a reduction in finite precision, equality of the ideal lenses alone gives no corresponding runtime guarantee.

5 Baselines and entropy weighting

5.1 Real-coded genetic algorithm

The genetic baseline uses a population of eight parameter vectors and ranks them by the regularized batch objective, using the same batch for all candidates in a generation. The two best candidates are copied to the next generation. For each of six offspring, two parents are drawn independently from the best four, and a uniformly drawn scalar gives an affine crossover between the parents. Independent normal mutations are scaled by parameter block. Every candidate evaluation uses the same distributed forward program as SGD. It sends no backward request. At a checkpoint the best individual of the latest evaluated generation is used. This is a specific real-coded genetic algorithm in the sense of selection, recombination and mutation [7].

5.2 Entropy-maximizing additive boosting

A weak atom $a_j(x)$ is one scalar feature, one frozen hidden unit, or the constant bias atom. A block $v_j \in \mathbb{R}^K$ controls the additive logits $a_j(x)v_j$. At each round, the baseline chooses one atom and adds a vector to its coefficient. Repeated selections add to the same coefficient, so the number of active atoms is bounded by the dictionary size. The linear and RBF tasks use the same hypothesis class as SGD. On the MLP, the weak dictionary freezes the hidden units at the matched initialization. We include SGD with the same hidden blocks frozen as a separate control.

Let u denote the uniform distribution on the B batch positions. Given the current losses ℓ_a, choose weights

$$q_\tau = \arg\max_{q \in \Delta_B} \left\{ \sum_a q_a \ell_a - \tau \, \mathrm{KL}(q\|u) \right\}, \qquad (1)$$

$$(q_\tau)_a = \frac{\exp(\ell_a/\tau)}{\sum_b \exp(\ell_b/\tau)}, \qquad \tau > 0.$$

This tradeoff assigns more weight to high-loss positions and rewards entropy. If one fixes the mean loss at $\sum_a (q_\tau)_a \ell_a$, then q_τ is the unique maximum-entropy distribution for that constraint. Indeed the expression in (1) equals

$$\tau \log \left(B^{-1} \textstyle\sum_a e^{\ell_a/\tau} \right) - \tau \, \mathrm{KL}(q\|q_\tau).$$

On the constraint set, maximizing it is the same as maximizing entropy. Its maximum value is a smooth batch risk, whose derivative is $\sum_a (q_\tau)_a \nabla \ell_a$. At fixed losses the derivative of $H(q_\tau)$ in τ is $\mathrm{Var}_{q_\tau}(\ell)/\tau^3$. Uniform weighting is the $\tau \to \infty$ control. Minimizing the smoothed batch risk is a different stochastic objective from minimizing the ordinary batch mean. In particular the nonlinear batch weights do not in general give an unbiased gradient for the corresponding full-training-set entropic risk.

With $p_a = \mathrm{softmax}(f_\theta(x_{i_a}))$ and e_y the one-hot vector, define

$$g_j = \lambda v_j + \sum_a (q_\tau)_a a_j(x_{i_a})(p_a - e_{y_{i_a}}),$$

$$D_j = .05 + \lambda + \sum_a (q_\tau)_a a_j(x_{i_a})^2.$$

Minimizing the block surrogate $g_j^\top \delta + D_j \|\delta\|^2/2$ would improve it by $\|g_j\|^2/(2D_j)$. The implementation chooses the largest $\|g_j\|^2/D_j$ and takes $-.75\, g_j/D_j$ on that block. The coordinator sends q_τ and the weighted logit residuals to both workers. Each returns its best local atom with score, divisor and gradient. The coordinator also considers the bias. A tie keeps the earlier candidate, with bias first, followed by worker 1 and worker 2, and feature index order within a worker. This damped surrogate is a selection rule. No claim of per-round monotonicity for the full training loss is needed. The method is related to stagewise boosting and entropy-regularized boosting [8, 9], with a different multiclass objective and update.

6 Three-guest experiment

6.1 Placement and message accounting

Every run boots three independent Alpine Linux 3.23.8 initramfs guests, one virtual CPU and 192 MiB each, on QEMU 10.0.13 with TCG [14]. The coordinator C owns labels, the global bias, parameter snapshots, the loss and optimizer state. Workers W_1, W_2 own feature shards and their local forward and reverse computations. The host builds the guests and relays Ethernet frames

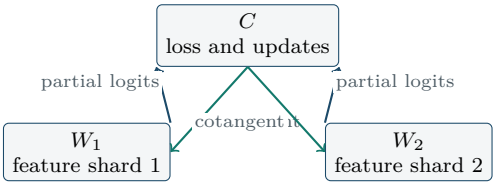


Figure 1: Primal outputs and their reverse seeds on three guests. Requests carry batch indices and the parameter snapshot. In the reverse round, the workers return their parameter-block pullbacks.

Table 1: Fixed training, validation and test splits. Counts refer to the data loaded by the guests.

Workload	Train	Val.	Test	Params.
Iris RBF	90	30	30	75
Digits RBF	1,074	355	368	810
Digits MLP	1,074	355	368	2,410
Fashion linear	4,000	1,000	2,000	500

on an isolated virtual LAN. No optimization or scoring runs on the host. Guests exchange TCP messages over emulated e1000 interfaces. Each worker records its own application-byte and operation counts, which the verifier reconciles against the coordinator's records.

Each worker emits $z_i \in \mathbb{R}^K$ per row. The coordinator uses $z = (b + z_1) + z_2$. For a linear shard the worker computes $z_i = W_i^\top x_i$ and its pullback accumulates $x_i \bar{z}^\top$. For the MLP it computes $z_i = V_i^\top \tanh(A_i x + b_i)$, with 16 hidden units per worker. Both MLP workers receive the same preprocessed inputs and own disjoint hidden/output blocks. For $h_i = \tanh(A_i x + b_i)$, its local formulas are

$$\bar{V}_i = h_i \bar{z}^\top, \qquad t_i = (V_i \bar{z}) \odot (1 - h_i^2),$$

$$\bar{A}_i = t_i x^\top, \qquad \bar{b}_i = t_i.$$

The coordinator sums its bias cotangents over rows and adds the ridge gradient once, after collecting all parameter blocks. The tape stores hidden activations and the snapshot used for each worker. Cross-guest messages carry a 20-byte header on requests and arrays of indices or binary64 values as appropriate. Bytes reported below are application payload and framing over both links, excluding the TCP/IP and Ethernet headers.

6.2 Datasets and hypotheses

Iris RBF. The three-class Iris task uses the corrected UCI rows [10]. Train/validation/test indices are fixed by a stratified 60/20/20 split. The four input measurements are standardized using training statistics. The head uses eight training landmarks per class, giving 24 Gaussian radial basis features.

Digits RBF and MLP. The 1,797 rows of the UCI Optical Recognition of Handwritten Digits test file [11] are split into new, disjoint 60/20/20 partitions by class. Thus these accuracies are for the supplied split of that

file. Inputs are 8-by-8 pixel counts divided by 16. The kernel model uses 80 training landmarks, eight per class. The neural model standardizes its 64 inputs with training statistics and learns 32 tanh units in two shards.

Fashion linear. We take 400 training and 100 validation images per class from the official Fashion-MNIST training set and 200 test images per class from its official test set [12]. Nonoverlapping 4×4 average pooling produces 49 features from each 28×28 grayscale image. Training-only standardization follows. The head is ten-class softmax with 49 features and a bias. This is an image classification workload with a deliberately small linear head.

The RBF feature for landmark c_j is $\exp(-\gamma\|x - c_j\|^2)$. The program sets γ to the reciprocal of the median squared distance of 512 reproducibly sampled training pairs. It standardizes each feature using training statistics. This realizes a finite kernel classifier with $K_{\mathrm{fin}}(x, x') = \langle \phi(x), \phi(x') \rangle$. It uses regularization in feature coefficient space. Kernel feature methods provide the relevant comparison class [13], while this construction uses fixed training landmarks. The archive records all split indices, landmark rows and bandwidths, together with the code that regenerates the feature shards. Table 1 gives the model sizes.

6.3 Budgets and selection protocol

We run each workload at seeds $11, 23, 37, 53, 71$ with the fixed splits. A configuration receives 384 forward queries of 48 examples each, or 18,432 example evaluations. GA spends eight queries on one common batch for each of 48 generations. Each other optimizer consumes 384 independently sampled batches and one forward query per batch. Thus the accounting matches forward exposure, including all GA candidate evaluations. It permits different independent draws, backward work and wall time. For all methods $\lambda = 10^{-4}$, and the same seed sets the common initial parameter vector. GA's remaining initial individuals perturb that vector. The two MLP boosting controls and frozen SGD share the same hidden-feature initialization as full SGD.

Each primary method has three hyperparameter choices. SGD starts at $\eta_0 \in \{.04, .12, .36\}$ and uses $\eta_s = \eta_0/\sqrt{1 + s/192}$ at zero-based step s. GA uses initial mutation scale in $\{.04, .12, .30\}$, annealed by $1 - .6s/384$ where s is the number of prior candidate evaluations. For the bias and hidden-bias entries the relative mutation scale is 1, for input weights it is the inverse square root of input width, and for output weights it is the inverse square root of total feature or hidden width. The entropy variants use $\tau \in \{.5, 1, 2\}$ and the uniform control has one configuration. No test metric enters model selection. For each method and seed we select the lowest validation log loss after query 384. Intermediate validation checkpoints at 96, 192 and 288 show the course of training. The test set is evaluated once per selected configuration. Evaluation, audit and tuning overhead are included in the cluster-wide counts and excluded from the per-configuration training budgets.

7 Results and replication

7.1 Derivative and schedule checks

Before training, each task tests a regularized objective on eight rows. For four unit directions (coordinator bias, each worker block, and all blocks), the coordinator computes central differences at radius 10^{-5} with distributed forward queries. It compares each estimate with the directional contraction of the distributed cotangent. The largest absolute difference is 3.83×10^{-11}. Reversing the two worker dispatch and collection orders, with fixed arithmetic order for reductions, changes neither the computed batch loss nor the returned gradient in these checks. This probes the implementation at finite points and is separate from the real-arithmetic proof.

In the reference run each worker recorded $87,460$ forward requests, $28,808$ full reverse requests and $30,720$ boosting statistics requests. The reconciled application traffic was 1.922 GiB across both links, including validation and audits. The virtual switch recorded $2,934,129$ Ethernet frames from three distinct guest MAC addresses. The second run booted fresh guests, built bit-identical guest binaries and initramfs images on the recorded toolchain, and reproduced all 860 validation checkpoints, 85 test records and four derivative audit records exactly apart from the excluded timing fields. The Ethernet frame counts differ between the runs.

7.2 Accuracy, loss and cost

Tables 2 and 3 give the primary comparisons. Full SGD has the lowest mean held-out log loss on all four workloads. It has the highest mean accuracy on both digit tasks and Fashion-MNIST and ties uniform boosting on Iris. The Fashion accuracy difference between SGD and entropy boosting is 5.29 percentage points. For Digits MLP, frozen SGD reaches 88.91% with mean log loss 0.380, versus 96.09% and 0.123 for full SGD. That control separates changing the hidden representation from changing the optimizer within a fixed dictionary. Frozen SGD uses the full reverse protocol and suppresses hidden updates, so it does not reduce the measured traffic.

Entropy weighting raises mean accuracy by 0.38 points on Digits MLP and 0.17 on Fashion, and lowers it by 0.67 on Iris and 0.27 on Digits RBF. Mean log loss is higher with entropy weighting on each workload. The mean normalized weighting entropy $H(q)/\log B$ at selected configurations ranges from 0.926 to 0.990. The accuracy gains on two tasks coexist with higher mean log loss.

The genetic baseline consumes the fewest application bytes and has the shortest selected-configuration median

Table 2: Held-out accuracy in percent. Entries are mean ± sample standard deviation across five seeds on fixed splits. The uniform column uses the same weak atoms as entropy boosting.

Workload	SGD	Genetic algorithm	Entropy boosting	Uniform boosting
Iris RBF	96.67±0.00	90.00±2.36	96.00±1.49	96.67±0.00
Digits RBF	95.38±0.43	43.75±2.72	93.21±0.86	93.48±0.43
Digits MLP	96.09±0.68	26.41±4.46	88.21±0.87	87.83±0.40
Fashion linear	78.24±0.77	33.87±4.12	72.95±0.69	72.78±0.57

Table 3: Mean test negative log likelihood in nats per example.

Workload	SGD	GA	Entropy	Uniform
Iris RBF	0.078	0.259	0.089	0.085
Digits RBF	0.147	1.646	0.317	0.314
Digits MLP	0.123	2.044	0.525	0.515
Fashion linear	0.621	1.863	0.893	0.829

Table 4: Median training traffic in MiB at the selected configuration. The count combines both worker links in both directions.

Workload	SGD	GA	Entropy	Uniform
Iris RBF	2.28	1.21	2.38	2.38
Digits RBF	10.48	5.31	8.49	8.49
Digits MLP	19.86	10.00	13.18	13.18
Fashion linear	8.67	4.40	7.58	7.58

training time on each workload in this run. For Digits MLP the selected medians are 2.60 s for full SGD, 1.94 s for GA and 2.38 s for entropy boosting. These durations describe the emulated single-host deployment and exclude evaluation. Table 4 shows that returning full gradients can be materially more expensive than returning an atom summary. In SGD the reverse round also returns a parameter block on each link. Boosting sends batch weights and returns only the best atom summary from each worker. On Iris, the compact full gradients make the two budgets similar.

Figure 2 shows the validation trajectories in forward queries. The fixed budget captures a regime in which GA has only 48 population updates. The best GA mutation scale and SGD learning rate often lie at the upper grid boundary, so the comparison describes these particular grids. It does not estimate the best attainable score for any algorithm. The five seeds reuse one split, so their spread measures sensitivity to initialization and sampling on that split. Iris has only 30 test examples. Its observed ties and one-example changes should be read at that resolution. The Fashion workload uses pooled pixels and a subset. It does not estimate accuracy for the standard full-resolution benchmark or a modern convolutional model.

8 Scope and implications

A reverse interface specifies more than the presence of derivatives. It specifies the cotangent space on each port, the snapshot on which the pullback depends, and the aggregation of all uses of a wire. A synchronous implementation can therefore change its legal schedule or placement while preserving the semantic update, provided it preserves those data. A cost analysis must still retain the trace of transmissions and placement. The same lens admits different deployments with different message traffic and runtime.

The experiments use model sharding over one emulated LAN. There is no comparison against multiple physical hosts, decentralized consensus, stale gradient protocols, or fault-tolerant recovery. The raw transcripts are generated by the provided program, so their hashes and counter reconciliation establish repeatability and internal consistency. The rebuild and repeat-run comparison provide a direct check of the reported results. A program with data-dependent branching can use a fixed executed trace for conditional differentiation, with separate conditions needed near switching boundaries. Feedback can be unrolled to a finite acyclic program at a fixed horizon. Neither construction licenses transposing a nondifferentiable branch decision or ignoring the state needed to reverse a loop.

9 Reproduction and disclosures

The companion archive includes the C implementation, a Python 3 standard-library guest builder and runner, SHA-256 pins, checksum-verified prepared data shards, both runs' serial logs and manifests, the verifier, and the double-column LaTeX source. On a Linux x86-64 host, QEMU system emulation and a C compiler with static libc suffice to build and run the experiments. The archive's README gives the commands for a full cluster run, the cross-run comparison and regeneration of the manuscript tables and curves. The data shards can be regenerated from the pinned upstream datasets. The verifier treats guest identities, Ethernet frame counts and wall-clock measurements as execution evidence whose values can vary, and checks the stable numerical and application-counter fields against the reference run. No accelerator or external service is used by the guests.

The commissioning researcher is the responsible contrib-

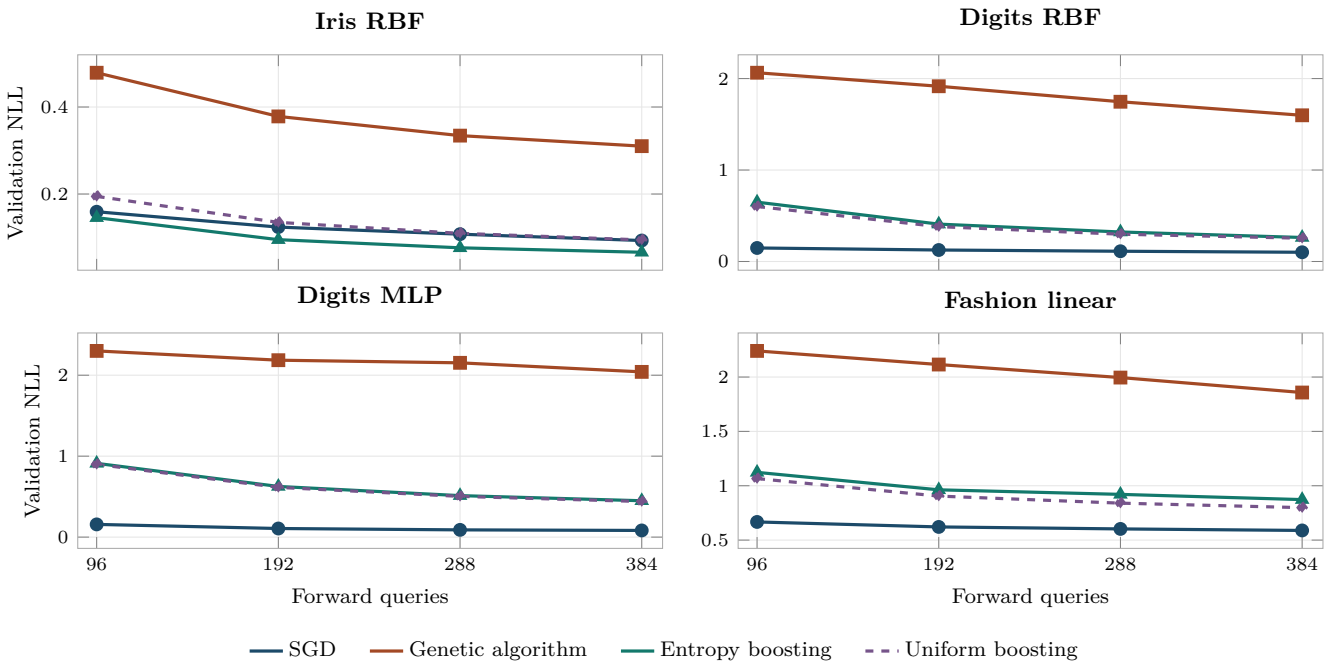


Figure 2: Mean validation negative log likelihood across five seeds at four query checkpoints, using the configuration selected by final validation loss for each seed. Each panel uses its own vertical scale. The curves are retrospective summaries of the selected configurations.

utor. The manuscript uses an anonymous byline. An AI coding and writing assistant developed the manuscript and the experimental software, ran the cluster twice, checked the quantitative results and assembled the replication artifacts under the researcher's direction. The mathematical results are supported by prose proofs.

References

[1] R. Cockett, G. Cruttwell, J. Gallagher, J.-S. P. Lemay, B. MacAdam, G. Plotkin, and D. Pronk. Reverse derivative categories. In *CSL 2020*, LIPIcs 152, article 18. doi:10.4230/LIPIcs.CSL.2020.18.

[2] G. S. H. Cruttwell, B. Gavranović, N. Ghani, P. Wilson, and F. Zanasi. Categorical foundations of gradient-based learning. In *ESOP 2022*, LNCS 13240, pp. 1–28. doi:10.1007/978-3-030-99336-8_1.

[3] J. Utke, L. Hascoët, C. Hill, P. Hovland, J. Riehme, and U. Naumann. Toward adjoinable MPI. In *PDSEC*, 2009. INRIA record.

[4] J. Lurie. *Higher Topos Theory*. Annals of Mathematics Studies 170, Princeton University Press, 2009. arXiv:math/0608040.

[5] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review* 60(2), pp. 223–311, 2018. doi:10.1137/16M1080173.

[6] T. Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics* 370, 107239, 2020. arXiv:1908.07021.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992. MIT Press edition.

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1), pp. 119–139, 1997. doi:10.1006/jcss.1997.1504.

[9] M. K. Warmuth, K. A. Glocer, and S. V. N. Vishwanathan. Entropy regularized LPBoost. In *ALT 2008*, LNCS 5254, pp. 256–271. doi:10.1007/978-3-540-87987-9_23.

[10] R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. doi:10.24432/C56C76. Corrected data file.

[11] E. Alpaydın and C. Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998. doi:10.24432/C50P49. Data file.

[12] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST, a novel image dataset for benchmarking machine learning algorithms, 2017. arXiv:1708.07747.

[13] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, 2007. NeurIPS proceedings.

[14] QEMU Project. *Direct Linux Boot* and *Network emulation*. Boot documentation and network documentation.