SCORE BROADCAST AND DECORRELATION: A GENERAL FRAMEWORK FOR BROADCAST-BASED CREDIT ASSIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce *Score Broadcast and Decorrelation* (SBD), a principled framework for broadcast-based credit assignment for general families of differentiable losses. Error broadcast is a biologically plausible alternative to backpropagation that sends output information to hidden layers without weight transport. The Error Broadcast and Decorrelation (EBD) framework, recently introduced for the mean-squared-error (MSE) setting, grounded this mechanism in the stochastic orthogonality of optimal estimators, under which the error of the optimal estimator is orthogonal to functions of the input. We generalize that foundation by introducing an orthogonality principle between the *output score* (the gradient of loss with respect to the final-layer output) and hidden-layer activations, which holds whenever the optimal score has conditional mean zero. This single principle unifies broadcast-based credit assignment across the standard differentiable-loss families, including cross-entropy, Bregman divergences (with MSE as a special case), proper scoring rules through unconstrained links, and exponential-family negative log-likelihoods. The framework supplies a theoretical grounding for the three-factor learning rule under general losses, with the neuromodulatory factor derived as the broadcast loss score rather than postulated. We derive the cross-entropy case explicitly, characterize the admissible loss class, and introduce a *score vector expansion* technique that enriches the broadcast signal while preserving the orthogonality framework. Experiments on CIFAR-10 and Tiny ImageNet show that SBD substantially improves over existing broadcast approaches, with score vector expansion delivering further gains. Overall, this work identifies the loss score as the signal to broadcast, supplies the orthogonality theory and theoretical grounding for the three-factor learning rule from neuroscience, and shows how score vector expansion enriches the decorrelation directions of the resulting objective.

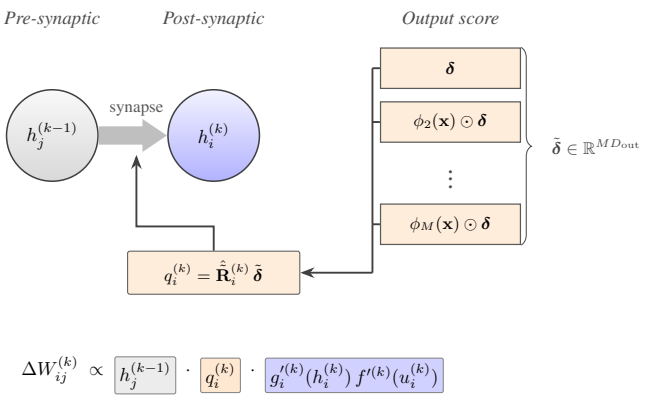
1 INTRODUCTION

Neural networks serve as the fundamental computational models of natural intelligence and form the core engine of modern AI. Across both domains, a key challenge is the credit assignment problem: how to update local synaptic weights to optimize a global performance metric. The dominant solution in machine learning is backpropagation (BP), which derives exact gradients by transmitting output errors backward through the network (Rumelhart et al., 1986). Although highly effective, BP requires symmetric backward pathways and exact weight transport, constraints long viewed as biologically implausible (Crick, 1989; Lillicrap et al., 2020).

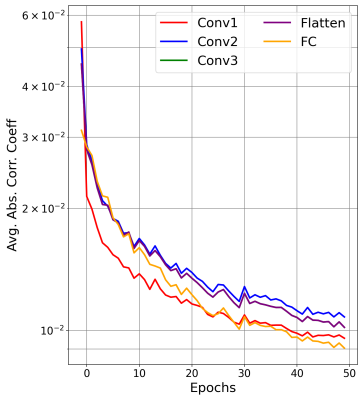
These constraints have driven the search for alternative credit assignment frameworks that relax the strict routing requirements of BP (Whittington and Bogacz, 2019). One notable family of such approaches relies on *error broadcast*: distributing global output information directly to hidden layers rather than propagating it sequentially. Although attractive and biologically plausible, this raises a fundamental question: *for a given loss, what specific quantity should be broadcasted, and what theoretical principle justifies that such a decentralized mechanism can drive learning?*

The *Error Broadcast and Decorrelation* (EBD) framework (Erdogan et al., 2025) provided a principled answer in the mean-squared-error (MSE) setting. Its starting point is the stochastic

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Three-factor SBD update with score-vector expansion. At layer k, the synaptic update uses presynaptic activity $h_j^{(k-1)}$, postsynaptic sensitivity $g_i'^{(k)}(h_i^{(k)})\, f'^{(k)}(u_i^{(k)})$, and a broadcast modulation $q_i^{(k)}$ obtained by projecting the expanded score vector $\tilde{\boldsymbol{\delta}} = [\boldsymbol{\delta};\, \phi_2(\mathbf{x}) \odot \boldsymbol{\delta};\, \ldots;\, \phi_M(\mathbf{x}) \odot \boldsymbol{\delta}]$, where $\boldsymbol{\delta} = \nabla_{\mathbf{a}} \mathcal{L}$ is the output score.



(b) Empirical correlations between the cross entropy output score and hidden-layer activations in a backpropagation-trained CIFAR-10 CNN. These correlations decrease during optimization, consistent with Theorem 2.

Figure 1: Illustrations of the SBD framework and the empirical score–activation orthogonality.

orthogonality property of MMSE estimation: at optimality, the estimation error is orthogonal to every input-measurable function whose product with the error is integrable. EBD turns this into layerwise decorrelation objectives between hidden activations and the broadcast output error, yielding local *three-factor learning rules*, a long-standing class of biologically motivated synaptic update rules, in which a presynaptic activity term and a postsynaptic sensitivity term are gated by a third, neuromodulatory factor (Frémaux and Gerstner, 2016; Gerstner et al., 2018; Kuśmierz et al., 2017; Schultz, 1998). However, this foundation is specific to squared error. In classification, the standard objective is cross entropy (CE), and more generally one wishes to optimize differentiable losses for which the MSE output error is not the natural error signal. A general theory of broadcast learning therefore requires a loss-dependent notion of what should be broadcast.

In this article, we identify that quantity as the *output score*, defined as the gradient of the loss with respect to the final layer output. For cross entropy, this score is $\boldsymbol{\delta} = \mathbf{p} - \mathbf{y}$. We show that, at the population cross entropy optimum, this score is conditionally mean zero and therefore orthogonal to any deterministic function of the input, including hidden layer activations. More generally, the same principle applies to differentiable losses whose conditional risks are characterized by a zero score condition. Thus, the MSE output error used by EBD and the cross entropy score used in classification are instances of the more general score-based orthogonality principle.

Building on this principle, we formulate *Score Broadcast and Decorrelation* (SBD), a broadcast-and-decorrelate framework for general differentiable losses that supplies a unified theoretical grounding for the three-factor learning rule introduced above, with the neuromodulatory factor *derived* as the broadcast loss score rather than postulated; Figure 1a depicts the resulting update, including the score-vector expansion of Section 6. The MSE-based EBD update is recovered as one specific instance.

We also introduce score vector expansion. The output score has dimension equal to the number of output coordinates, which can limit the number of independent decorrelation directions available to wide hidden layers. SBD can expand the broadcast vector by multiplying the score with deterministic modulators, such as functions of the predictive distribution. These modulated scores preserve the conditional mean zero property at the population optimum while providing a richer set of decorrelation directions.

We evaluate SBD on CIFAR-10 and Tiny ImageNet using CNN architectures matched to the original EBD setting. The experiments are intended as controlled proof of concept tests within broadcast learning rather than as claims of superiority over exact BP. BP is included as a reference optimizer, while the main comparisons are to MSE based EBD and other broadcast baselines. In this setting, SBD improves over other broadcast formulations, and score vector expansion provides additional gains.

1.1 Related work

Alternatives to BP span several major families, including *contrastive methods* such as Equilibrium Propagation (Scellier and Bengio, 2017; 2019), *target propagation* (Bengio, 2014; Lee et al., 2015), *forward-only methods* (Hinton, 2022), *layerwise local losses* (Nøkland and Eidnes, 2019), *predictive methods* (Rao and Ballard, 1999; Whittington and Bogacz, 2017; Golkar et al., 2022), *similarity matching* (Qin et al., 2021), *feedback alignment* (Lillicrap et al., 2016), and *learned feedback pathways* (Akrout et al., 2019).

Most closely related to the present work are *error broadcast* methods, which send global output information directly to hidden layers. Direct Feedback Alignment (DFA) projects the output loss derivative through fixed random feedback matrices (Nøkland, 2016; Bartunov et al., 2018; Han and Yoo, 2019; Launay et al., 2019; 2020; Bordelon and Pehlevan, 2023). Although this removes weight transport, DFA does not derive the broadcast signal from a loss-dependent population-optimality principle or provide a mechanism for expanding the broadcast beyond the output dimension. Clark et al.'s Global Error Vector Broadcasting (GEVB) method (Clark et al., 2021) uses a global error vector to modulate local plasticity in vectorized networks with nonnegative weights beyond the first layer, but does not establish a general score-orthogonality principle or conditions under which decorrelation is sufficient for population optimality. Most directly, Error Broadcast and Decorrelation (EBD) (Erdogan et al., 2025) gives broadcast learning a normative foundation by deriving layerwise decorrelation objectives from orthogonality of the MMSE estimation error, but this foundation is restricted to squared loss. EBD also reports an empirical decrease in output-error–activation correlations under cross-entropy backpropagation and discusses additional nonlinear feature constraints, without developing the general-loss framework here.

1.2 Contributions

- **The output score as the broadcast signal.** We identify the loss score $\boldsymbol{\delta} = \nabla_{\mathbf{a}}\mathcal{L}(\mathbf{y}, \mathbf{a})$, taken with respect to an unconstrained output parameter $\mathbf{a}$, as the principled quantity to broadcast. This choice recovers the EBD output error under MSE and yields $\mathbf{p} - \mathbf{y}$ under softmax cross entropy.
- **A characterization that makes decorrelation a training criterion.** We characterize when the population-optimal score has zero conditional mean and is therefore orthogonal to every input-measurable feature. Conversely, when the conditional risk is convex in the unconstrained output parameter, a conditionally mean-zero score identifies a pointwise population minimizer; score decorrelation is thus an optimality criterion rather than merely a diagnostic.
- **A broad loss family and a derived three-factor rule.** The characterization covers Bregman divergences (including MSE), proper scoring rules through unconstrained links (including cross entropy), and exponential-family negative log-likelihoods. Across these losses, SBD derives the neuromodulatory factor in the local three-factor learning rule from the task loss itself rather than postulating it.
- **A remedy for the broadcast rank bottleneck.** We introduce score vector expansion (SVE), which stacks deterministically modulated score blocks. SVE preserves conditional-mean-zero orthogonality at the same population optimum while increasing the broadcast dimension and allowing additional decorrelation directions.
- **Empirical validation.** On CIFAR-10 and Tiny ImageNet, SBD substantially improves over broadcast baselines, and SVE provides further gains. Update-alignment diagnostics also show positive cosine similarity with BP gradients throughout training.

2 A review of error broadcast and decorrelation method

We study a supervised learning problem using a multilayer perceptron with L layers, including the output layer. Let $\mathbf{h}^{(k)} \in \mathbb{R}^{N^{(k)}}$ denote the activation vector at layer k, where $\mathbf{h}^{(0)} = \mathbf{x}$ is the input and $\mathbf{h}^{(L)}$ is the output. For each layer $k = 1, \ldots, L$, the pre-activation and activation are written as

$$\mathbf{u}^{(k)} = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}, \qquad \mathbf{h}^{(k)} = f^{(k)}\left(\mathbf{u}^{(k)}\right),$$

where $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$ are the weights and biases of layer k, and $f^{(k)}(\cdot)$ is its activation function.

EBD (Erdogan et al., 2025) is a broadcast-based alternative to backpropagation grounded in the stochastic orthogonality property of nonlinear MMSE estimators. If $\widehat{\mathbf{y}}^\star(X) = \mathbb{E}[\mathbf{Y} \mid X]$ is the optimal MMSE estimator, then its estimation error $\boldsymbol{\epsilon}^\star = \mathbf{Y} - \widehat{\mathbf{y}}^\star(X)$ is orthogonal to every measurable function $g(X)$ for which $\mathbb{E}[\|g(X)\boldsymbol{\epsilon}^{\star T}\|_F] < \infty$. EBD chooses these functions as hidden-layer activations or their nonlinear transforms and, using the network convention of Erdogan et al. (2025), defines the current MSE output error as $\boldsymbol{\epsilon} := \mathbf{h}^{(L)} - \mathbf{y}$. At layer k, EBD seeks to enforce

$$\mathbf{R}_{g\epsilon}^{(k)} = \mathbb{E}\big[g^{(k)}(\mathbf{h}^{(k)})\boldsymbol{\epsilon}^T\big] = \mathbf{0},$$

where $g^{(k)}(\mathbf{h}^{(k)})$ is a nonlinear function of the layer activations whose outer product with $\boldsymbol{\epsilon}$ is integrable. Because $\mathbf{h}^{(k)}$ is itself a deterministic function of the input, this is a valid choice of input-measurable function; typically, $g^{(k)}(\mathbf{h}^{(k)}) = \mathbf{h}^{(k)}$. The autoregressive estimate $\widehat{\mathbf{R}}_{g\epsilon}^{(k)}[m]$ for minibatch m of size B is defined in Appendix C.1. EBD minimizes the layerwise-defined decorrelation objective

$$\mathcal{J}_{\mathrm{EBD}}^{(k)}[m] = \frac{1}{2}\left\|\widehat{\mathbf{R}}_{g\epsilon}^{(k)}[m]\right\|_F^2.$$

Thus the stochastic orthogonality condition is turned into layer-specific loss functions that decorrelate transformed layer activations from the broadcast output error (Erdogan et al., 2025).

Differentiating this loss gives two gradient components: one contains derivatives of the layer activations with respect to the layer parameters, whereas the other contains derivatives of the output error and requires signals propagated from deeper layers. EBD retains the activation-derivative component and defines the projected output error

$$\mathbf{q}^{(k)}[n] = \widehat{\mathbf{R}}_{g\epsilon}^{(k)}[m]\,\boldsymbol{\epsilon}[n], \qquad n = mB+1,\ldots,(m+1)B,$$

where n is the sample index. The resulting single-sample hidden-layer update, for $B = 1$, is

$$\Delta W_{ij}^{(k)}[n] = \zeta\, g_i'^{(k)}\Big(h_i^{(k)}[n]\Big)\, f'^{(k)}\Big(u_i^{(k)}[n]\Big)\, q_i^{(k)}[n] h_j^{(k-1)}[n].$$

Here ΔW is a parameter increment and $\zeta = -\eta(1-\lambda)$ for gradient descent with learning rate $\eta > 0$. This update has the three-factor structure in neuroscience: a presynaptic activity, a postsynaptic factor formed by the derivatives of $g^{(k)}$ and $f^{(k)}$, and the projected global error as the modulatory factor. EBD's derivation, however, is tied to the MSE loss only. The proposed SBD framework generalizes the derivation of the three-factor rule to a wider family of losses, where MSE is a special instance.

3 Extension to cross entropy loss

We now extend the EBD framework from MSE to multiclass classification under cross entropy loss. Let $(X, \mathbf{Y}) \sim \mathbb{P}_{X,\mathbf{Y}}$ be jointly distributed input–label pairs taking values in $\mathcal{X} \times \{\mathbf{e}_1, \ldots, \mathbf{e}_D\}$, where $\mathbf{e}_d \in \mathbb{R}^D$ is the d-th canonical basis vector. As before, the network produces hidden activations $\mathbf{h}^{(k)}$ for $k = 1, \ldots, L-1$, and the output layer produces logits

$$\mathbf{a} = \mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}, \tag{1}$$

with corresponding predicted probabilities $\mathbf{p} = \mathrm{softmax}(\mathbf{a})$, and cross entropy loss

$$\mathcal{L}_{\mathrm{CE}}(\mathbf{y}, \mathbf{a}) = -\sum_{d=1}^{D} y_d \log p_d.$$

We refer to the gradient with respect to the logits $\mathbf{a}$ as the *output score*, which here takes the form

$$\boldsymbol{\delta}_{\mathrm{CE}} = \nabla_{\mathbf{a}}\mathcal{L}_{\mathrm{CE}}(\mathbf{y}, \mathbf{a}) = \mathbf{p} - \mathbf{y}.$$

We will show that this cross entropy score plays the role of the output error $\boldsymbol{\epsilon}$ in the MSE version of EBD, satisfying the analogous orthogonality property at population optimality.

To obtain the cross entropy analog of MMSE orthogonality, define

$$\mathbf{q}(\mathbf{x}) = \big[\ \mathbb{P}(\mathbf{Y} = \mathbf{e}_1 \mid X = \mathbf{x}), \ldots, \mathbb{P}(\mathbf{Y} = \mathbf{e}_D \mid X = \mathbf{x})\ \big]^T,$$

as the true conditional class-probability vector. For any predictor $\mathbf{p}(\mathbf{x})$, the cross entropy risk is

$$\mathcal{R}_{\mathrm{CE}}(\mathbf{p}) = \mathbb{E}\Big[-\sum_{d=1}^{D} Y_d \log p_d(X)\Big] = \mathbb{E}_X\big[H(\mathbf{q}(X), \mathbf{p}(X))\big],$$

where $H(\mathbf{q}, \mathbf{p})$ is the cross entropy between the true and predicted conditional distributions. Using the standard decomposition $H(\mathbf{q}, \mathbf{p}) = H(\mathbf{q}) + \mathrm{KL}(\mathbf{q}\|\mathbf{p})$, the unique population minimizer over probability predictors is $\mathbf{p}^\star(\mathbf{x}) = \mathbf{q}(\mathbf{x})$ almost surely, which yields the following identity replacing the MMSE estimation-error property.

Theorem 1 (Conditional mean-zero property for cross entropy) *Let $\mathbf{p}^\star(X)$ denote the population minimizer of the cross entropy risk. Then*

$$\mathbb{E}\big[\mathbf{p}^\star(X) - \mathbf{Y} \mid X\big] = \mathbf{0}.$$

The proof is deferred to Appendix A.1.

The conditional mean-zero identity becomes unconditional orthogonality with respect to arbitrary measurable functions of X via the tower property of conditional expectation.

Lemma 1 (Tower-property orthogonality) *Let $\mathbf{U}$ be an integrable random vector and let $g(X)$ be any measurable function of X such that $\mathbf{U}\, g(X)^T$ is integrable, i.e., $\mathbb{E}\big[\|\mathbf{U}\, g(X)^T\|\big] < \infty$. If $\mathbb{E}[\mathbf{U} \mid X] = \mathbf{0}$, then $\mathbb{E}\big[\mathbf{U}\, g(X)^T\big] = \mathbf{0}$.*

Applying the lemma with $\mathbf{U} = \mathbf{p}^\star(X) - \mathbf{Y}$ yields the main cross entropy orthogonality theorem.

Theorem 2 (Cross entropy orthogonality theorem) *Assume that the predictor class contains the true conditional distribution, so that $\mathbf{p}^\star(X) = \mathbf{q}(X)$. Then for any measurable function $g(X)$ such that $(\mathbf{p}^\star(X) - \mathbf{Y})\, g(X)^T$ is integrable, in particular, for any g with $\mathbb{E}\|g(X)\| < \infty$, since $\mathbf{p}^\star(X) - \mathbf{Y}$ is bounded,*

$$\mathbb{E}\big[(\mathbf{p}^\star(X) - \mathbf{Y}) g(X)^T\big] = \mathbf{0}.$$

In particular, choosing $g(X) = g^{(k)}(\mathbf{h}^{(k)}(X))$ gives

$$\mathbb{E}\big[(\mathbf{p}^\star(X) - \mathbf{Y}) g^{(k)}(\mathbf{h}^{(k)}(X))^T\big] = \mathbf{0}.$$

The proof is deferred to Appendix A.3.

Figure 1b illustrates this empirically: although the theorem is a population statement, the measured score–hidden correlations in a CNN trained with cross entropy steadily decline during backpropagation, indicating that optimization drives the network toward the predicted orthogonality regime.

Using this cross entropy counterpart of orthogonality, we define the layerwise decorrelation condition

$$\mathbf{R}_{\mathrm{CE}}^{(k)} = \mathbb{E}\big[g^{(k)}(\mathbf{h}^{(k)})(\mathbf{p} - \mathbf{y})^T\big] \in \mathbb{R}^{N^{(k)} \times D}, \qquad \mathbf{R}_{\mathrm{CE}}^{(k)} = \mathbf{0},$$

with objective $\mathcal{J}_{\mathrm{CE}}^{(k)} = \frac{1}{2}\|\mathbf{R}_{\mathrm{CE}}^{(k)}\|_F^2$. The corresponding local three-factor update is obtained by substituting $\boldsymbol{\delta} = \mathbf{p} - \mathbf{y}$ into the general SBD rule in Section 4; the cross-entropy implementation is detailed in Appendix C.

Sufficiency under a dense-feature assumption. Theorem 2 shows that score–feature orthogonality is *necessary* at the population cross entropy optimum. Sufficiency requires a sufficiently rich feature family, as discussed for general losses in Section 4. The finite-width algorithm should be viewed as an approximation to this idealized case. The detailed cross-entropy argument is in Appendix A.4.

4 Extension to more general losses: score broadcast and decorrelation

The cross entropy derivation reveals the central idea behind the generalized framework: what matters is not the MSE output error or the particular form of the cross entropy score, but the score of the

loss with respect to the network output. For a general differentiable loss $\mathcal{L}(\mathbf{y}, \mathbf{a})$ with $\mathbf{a} \in \mathbb{R}^{D_{\text{out}}}$, we define the score by

$$\boldsymbol{\delta}(\mathbf{y}, \mathbf{a}) := \nabla_{\mathbf{a}} \mathcal{L}(\mathbf{y}, \mathbf{a}).$$

This provides a universal broadcast signal. In particular, $\boldsymbol{\delta}_{\text{MSE}} = \mathbf{a} - \mathbf{y}$, and, $\boldsymbol{\delta}_{\text{CE}} = \mathbf{p} - \mathbf{y}$, so both MSE and cross entropy fit naturally into the same score-based formulation. For arbitrary differentiable losses, the first universal orthogonality statement arises from stationarity of the last-layer weights. Let $\mathbf{a}$ denote network output as in Eq. (1), define the population risk $\mathcal{R} = \mathbb{E}\big[\mathcal{L}(\mathbf{Y}, \mathbf{a}(\mathbf{X}))\big]$. Then differentiation with respect to the last-layer weight matrix gives

$$\nabla_{\mathbf{W}^{(L)}} \mathcal{R} = \mathbb{E}\big[\boldsymbol{\delta}\, \mathbf{h}^{(L-1)T}\big].$$

Consequently, every stationary point of the population risk satisfies the following proposition.

Proposition 1 (Stationary score-feature orthogonality) *Assume that $\mathcal{L}(\mathbf{y}, \mathbf{a})$ is differentiable in $\mathbf{a}$ and that gradient and expectation may be interchanged in the expression for $\mathcal{R}$. If $\mathbf{W}^{(L)}$ is stationary while all earlier parameters are fixed, then $\mathbb{E}\big[\boldsymbol{\delta}\, \mathbf{h}^{(L-1)T}\big] = \mathbf{0}$.*

The proof is deferred to Appendix A.5.

The proposition only covers the last hidden layer. Orthogonality with arbitrary measurable input functions requires the conditional mean-zero score property of the following theorem:

Theorem 3 (General score orthogonality theorem) *Let $\mathbf{a}^\star(X)$ be an optimal predictor at the population level for a differentiable loss $\mathcal{L}(\mathbf{y}, \mathbf{a})$, and let $\boldsymbol{\delta}^\star = \nabla_{\mathbf{a}} \mathcal{L}(\mathbf{Y}, \mathbf{a}^\star(X))$ be the corresponding optimal score. Assume that the optimal score satisfies the conditional mean-zero property $\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}$. Then for every measurable function $g(X)$ such that $\boldsymbol{\delta}^\star\, g(X)^T$ is integrable (equivalently, $\mathbb{E}\big[\|\boldsymbol{\delta}^\star\, g(X)^T\|\big] < \infty$),*

$$\mathbb{E}\big[\boldsymbol{\delta}^\star g(X)^T\big] = \mathbf{0}.$$

In particular, taking $g(X) = g^{(k)}(\mathbf{h}^{(k)}(X))$ yields layerwise score orthogonality.

The proof is deferred to Appendix A.6.

Remark. The conditional mean-zero hypothesis assumed in this theorem is broadly satisfied: in Section 5 and Appendix B we show that it holds whenever the conditional risk has an interior minimizer in an open parameter domain. This characterization covers the standard differentiable-loss families used in supervised learning, including Bregman divergences, proper scoring rules through unconstrained links, and exponential-family negative log-likelihoods. MSE and cross-entropy are canonical instances of the first and second families, respectively.

Sufficiency under a dense-feature assumption. The dense-feature argument depends on the loss only through its score. Let $\mathbf{a}(X)$ denote a *candidate* predictor (not assumed optimal) and let $\boldsymbol{\delta} := \nabla_{\mathbf{a}} \mathcal{L}(\mathbf{Y}, \mathbf{a}(X))$ be its score, assumed square-integrable. By the tower property, decorrelating $\boldsymbol{\delta}$ against a feature family whose linear span is dense in $L^2(P_X)$ forces $\mathbb{E}[\boldsymbol{\delta} \mid X] = \mathbf{0}$ almost surely. Under the regularity assumptions of Section 5, this identifies a pointwise minimizer whenever the conditional risk and its domain are convex. The cross-entropy case of Section 3 and the MSE case of EBD (Erdogan et al., 2025, Appendix B.2) are two instances; under SBD the same argument applies uniformly across the differentiable-loss families covered by the conditional-risk characterization.

Let d_k denote the dimension of $g^{(k)}(\mathbf{h}^{(k)})$. The resulting generalized feature-by-score correlation and score-broadcast objective are

$$\mathbf{R}^{(k)}_{\text{Score}} := \mathbb{E}\big[g^{(k)}(\mathbf{h}^{(k)})\boldsymbol{\delta}^T\big] \in \mathbb{R}^{d_k \times D_{\text{out}}}, \qquad \mathcal{J}^{(k)}_{\text{Score}} = \frac{1}{2}\left\|\mathbf{R}^{(k)}_{\text{Score}}\right\|_F^2,$$

with minibatch estimator

$$\widehat{\mathbf{R}}^{(k)}_{\text{Score}}[m] = \lambda \widehat{\mathbf{R}}^{(k)}_{\text{Score}}[m-1] + (1-\lambda) B^{-1}\, \mathbf{G}^{(k)}[m] \boldsymbol{\Delta}[m]^T,$$

where $\mathbf{G}^{(k)}[m] \in \mathbb{R}^{d_k \times B}$ collects the hidden features and $\boldsymbol{\Delta}[m] \in \mathbb{R}^{D_{\text{out}} \times B}$ collects the score vectors $\boldsymbol{\delta}[mB+1], \ldots, \boldsymbol{\delta}[mB+B]$.

As shown in Appendix C.2, differentiating $\mathcal{J}^{(k)}_{\mathrm{Score}}[m]$ gives one gradient component containing derivatives of the layer activations and another containing derivatives of the output score. The latter would require propagating signals through deeper layers, so the practical broadcast rule retains the activation-derivative component. The layerwise broadcast is

$$\mathbf{q}^{(k)}_{\mathrm{Score}}[n] = \widehat{\mathbf{R}}^{(k)}_{\mathrm{Score}}[m]\,\boldsymbol{\delta}[n], \qquad n = mB+1,\dots,(m+1)B,$$

and for $B = 1$ and elementwise $g^{(k)}$, the weight update keeps the generic local form

$$\Delta W^{(k)}_{ij}[n] = \zeta\, g_i'^{(k)}\Big(h^{(k)}_i[n]\Big)\, f'^{(k)}\Big(u^{(k)}_i[n]\Big)\, q^{(k)}_{\mathrm{Score},i}[n] h^{(k-1)}_j[n].$$

The update rule above provides a derivation of the three-factor learning rule, a long-standing model of synaptic plasticity in computational neuroscience (Frémaux and Gerstner, 2016; Kuśmierz et al., 2017), under a broad family of differentiable losses. In the neuroscience literature the neuromodulatory factor is typically postulated; under SBD it is derived *from the loss itself* as the broadcast loss score. Because the derivation depends on the loss only through its score, the same three-factor update covers every loss to which the conditional-mean-zero characterization applies, including Bregman divergences, exponential-family negative log-likelihoods, and proper scoring rules through unconstrained links. SBD is thus a unified derivation of the three-factor rule across the standard differentiable losses of supervised learning, with MSE-based EBD recovered as one instance. The CNN experiments additionally use activation-covariance regularization (Erdogan et al., 2025; Ozsoy et al., 2022; Bozkurt et al., 2023a) and layer-specific regularizers described in Appendices E.1 and F.

5 When does an unconstrained loss parametrization yield a conditionally mean-zero score?

Theorem 3 shows that full layerwise orthogonality follows once the population-optimal score satisfies $\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}$. We now ask when this hypothesis holds. The key step is to forget the network parameterization and study the loss as a function of the prediction variable alone.

For each input value x, define the conditional risk

$$C_x(\mathbf{a}) = \mathbb{E}\big[\mathcal{L}(\mathbf{Y},\mathbf{a}) \mid X = x\big],$$

where $\mathbf{a}$ is the output parameter with respect to which the SBD score $\boldsymbol{\delta} = \nabla_{\mathbf{a}}\mathcal{L}(\mathbf{Y},\mathbf{a})$ is taken, and assume that $\mathbf{a}$ ranges over an open set $\mathcal{A} \subseteq \mathbb{R}^m$. Losses with constrained natural prediction variables (e.g., probability losses on $\mathbf{p} \in \Delta_D$) are brought into this setting by composing with an unconstrained link $\mathbf{p} = \psi(\mathbf{a})$ and writing $\mathcal{L}(\mathbf{Y},\mathbf{a}) = \ell(\mathbf{Y},\psi(\mathbf{a}))$. The following theorem then characterizes when the conditional-mean-zero score property holds.

Theorem 4 (Conditional-risk characterization, informal) *Suppose C_x has an interior minimizer $\mathbf{a}^\star(x) \in \mathcal{A}$ for almost every x, with mild regularity allowing differentiation under the conditional expectation. Then the optimal score satisfies*

$$\mathbb{E}\big[\nabla_{\mathbf{a}}\mathcal{L}(\mathbf{Y},\mathbf{a}^\star(X)) \mid X\big] = \mathbf{0}.$$

If, in addition, $\mathcal{A}$ and C_x are convex for almost every x, the converse also holds: any predictor whose score has zero conditional mean almost surely minimizes C_x pointwise.

The full statement of the theorem and its proof are deferred to Appendix B (Theorem 5). Appendix B applies the theorem to three canonical loss families: Bregman divergences (with MSE as the special case $\phi(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|_2^2$), proper scoring rules through unconstrained links (with softmax cross-entropy as the canonical instance), and exponential-family negative log-likelihoods (where the conditional-mean-zero property reduces to moment matching). The main takeaway is that SBD subsumes the standard differentiable losses of supervised learning under a single principle, with MSE-based EBD recovered as one instance.

6 Score vector expansion

The output score $\boldsymbol{\delta}(\mathbf{Y},\mathbf{a})$ has dimension D_{out}, which is typically much less than the hidden layer width d_k. We use D_ε for the dimension of the signal broadcast to hidden layers: $D_\varepsilon = D_{\mathrm{out}}$ for

plain SBD and $D_\varepsilon = MD_{\text{out}}$ when M score blocks are stacked. Hence the unexpanded layerwise correlation matrix has rank at most D_{out}, and this bottleneck limits the number of independent decorrelation constraints imposed by SBD on the hidden representation. This limitation can be relaxed by enlarging the broadcast signal with deterministic modulations that preserve the conditional mean-zero property. The modulator need only be a function of X. Given any modulator $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^{D_{\text{out}}}$, define the modulated score at the population optimum by

$$\boldsymbol{\eta}^{\boldsymbol{\phi},\star}(X) \;:=\; \boldsymbol{\phi}(X) \odot \boldsymbol{\delta}\big(\mathbf{Y}, \mathbf{a}^\star(X)\big).$$

When the modulated score is integrable, it also has zero conditional mean because $\boldsymbol{\phi}(X)$ pulls out of the conditional expectation. We stack M such modulated scores into the expanded broadcast vector

$$\tilde{\boldsymbol{\delta}}(X) \;:=\; \big[\boldsymbol{\phi}_1(X) \odot \boldsymbol{\delta}(\mathbf{Y}, \mathbf{a}(X));\; \ldots;\; \boldsymbol{\phi}_M(X) \odot \boldsymbol{\delta}(\mathbf{Y}, \mathbf{a}(X))\big] \in \mathbb{R}^{D_\varepsilon}, \qquad D_\varepsilon = MD_{\text{out}}.$$

At the population optimum, this preserves layerwise orthogonality whenever the score–feature products are integrable, allowing additional decorrelation directions at linear broadcast cost (Appendix C.4). Output-dependent choices $\boldsymbol{\phi}(X) = \psi(\mathbf{a}(X))$ are a special case.

The general construction only requires the modulators $\boldsymbol{\phi}_\ell$ to be deterministic functions of X, enabling substantial freedom in their design; a natural and computationally inexpensive choice is to build them from quantities already produced by the forward pass, such as the predictive distribution $\mathbf{p}(X)$, which is itself a nonlinear deterministic function of X. Following this guideline, for the CNN experiments in Section 7 we adopt the $3D$-dimensional instance $\tilde{\boldsymbol{\delta}} = \big[\boldsymbol{\delta};\; \mathbf{p}(X) \odot \boldsymbol{\delta};\; \mathrm{roll}_k(\mathbf{p}(X)) \odot \boldsymbol{\delta}\big] \in \mathbb{R}^{3D}$, which augments the raw score with a confidence-weighted score block and a cyclic permutation of the probability coordinates. We use $k = 5$ in the reported experiments. Because class indices are arbitrary, k has no semantic interpretation. These blocks are one admissible choice among many; Appendix E.5 reports an ablation over alternatives, and more principled, data-adaptive constructions are left as future work. The numerical experiments of Section 7, most clearly the 3.1-point gain on Tiny ImageNet, show that this selected instance improves over unexpanded SBD. Full derivations and the expanded algorithm are deferred to Appendix D.

7 Numerical experiments

Because our main contribution is conceptual and theoretical, we designed our experiments as tests of the score broadcast principle within broadcast learning. BP with cross entropy is included as a reference optimizer; the main comparisons are to MSE based EBD and DFA based broadcast baselines. For CIFAR-10 (Krizhevsky, 2009), we use the CNN architecture and training setup of Erdogan et al. (2025), replacing MSE by cross entropy where appropriate. Table 1 shows that among broadcast methods, SBD with score expansion (SBD Exp) performs best: SBD improves on EBD from 66.4% to 69.2%, and score expansion further improves performance to 70.0%. A $4\times$ width run raises BP to 83.1% and SBD Exp to 74.5%. Training settings are given in Appendix E. Appendix E.6 also reports positive layerwise cosine similarity between raw local SBD directions and exact BP gradients on the same minibatches.

Table 1: CIFAR-10 test accuracy (%). In-house CE results report mean $\pm$ standard deviation over five runs and share the CNN architecture of Appendix E. MSE results are from Erdogan et al. (2025); NN-GEVB is a literature reference from Clark et al. (2021) using a different architecture.

BP (MSE)	DFA (MSE)	EBD (MSE)	BP (CE)	DFA (CE)	NN-GEVB (CE)	SBD (CE, Ours)	SBD Exp (CE, Ours)
75.2 ± 0.3	58.4 ± 1.6	66.4 ± 0.4	$\mathbf{78.5 \pm 0.5}$	65.3 ± 1.2	66.26	69.2 ± 0.7	$\underline{70.0 \pm 0.7}$

For Tiny ImageNet (Le and Yang, 2015), we use the 200 class, 6 layer CNN setup described in Appendix F. Table 2 shows the same qualitative ordering on a harder benchmark. BP remains the strongest reference optimizer, while SBD improves substantially over DFA and EBD (MSE). Score expansion again gives the best broadcast result. Across both benchmarks, SBD improves on MSE-based EBD, with further gains from the expanded variants. Beyond cross-entropy, Appendix B.1 reports Poisson regression with a prediction-bin calibration diagnostic motivated by the conditional-mean-zero property. Appendices E–H provide the experimental details.

Table 2: Tiny ImageNet accuracy (%; mean ± standard deviation over five runs), evaluated on the official validation set, used as the test set here. BP, DFA, and SBD use the CNN architecture of Appendix F. NN-GEVB is an in-house run using the code of Clark et al. (2021).

BP (MSE)	EBD (MSE)	BP (CE)	DFA (CE)	NN-GEVB (CE, in-house)	SBD (CE, Ours)	SBD Exp (CE, Ours)
36.0 ± 0.7	18.5 ± 0.7	$\mathbf{39.9 \pm 0.3}$	17.5 ± 0.4	18.8 ± 0.5	28.3 ± 0.4	$\underline{31.4 \pm 0.4}$

We further evaluate SBD in two complementary settings: deep residual architectures and deep recurrent networks with biologically plausible plasticity. On a wide, 20-layer residual network, the extended SBD variant described in Appendix H achieves $(75.90 \pm 0.60)\%$ test accuracy. A BP reference on the same wide architecture achieves $(94.50 \pm 0.09)\%$. This experiment extends the empirical scope of SBD to deep residual architectures, while the remaining gap to BP highlights an important direction for improvement. We also consider Correlative Information Maximization (CorInfoMax) networks, introduced by Bozkurt et al. (2023a) and subsequently trained with EBD by Erdogan et al. (2025). These networks derive their recurrent representation dynamics from a correlation-based information-maximization objective, combining interlayer prediction with adaptive lateral interactions. Here, CE-based score broadcasts supply task supervision through local plasticity while retaining CorInfoMax's recurrent inference. Training is fully online (batch size 1), using local Hebbian, anti-Hebbian, and three-factor updates. The LC architecture additionally uses spatially local, unshared weights, avoiding convolutional weight tying. On CIFAR-10, the 10-layer FC network achieves $(60.24 \pm 0.46)\%$, compared with the previously reported $(54.89 \pm 0.58)\%$ for EBD, while the LC network with nine hidden stages reaches $(71.98 \pm 0.49)\%$ (Appendix G). Together, these results demonstrate learning at depth both in residual architectures and under the combined constraints of recurrent lateral interactions, single-sample training, and local synaptic plasticity.

8 Conclusions and limitations

We introduced *Score Broadcast and Decorrelation* (SBD), a loss- and parameterization-aware framework for broadcast credit assignment. Under the stated conditions, the population-optimal output score is conditionally mean zero and hence orthogonal to input-measurable features, including hidden activations. When the conditional risk is convex in the chosen output coordinates, this conditional-mean-zero property also characterizes optimal predictions. SBD thereby provides a principled answer to what should be broadcast and a theoretical foundation for local three-factor learning: the modulatory signal is derived from the task loss rather than postulated, with MSE-based EBD recovered as a special case. Score vector expansion provides a second advance: deterministic input-dependent modulation preserves population-optimal orthogonality while allowing independent decorrelation directions beyond the raw output dimension, thereby relaxing an intrinsic rank bottleneck of unexpanded broadcast. Experiments on CIFAR-10 and Tiny ImageNet show improvements over the evaluated broadcast baselines and further gains from expansion. The extended ResNet experiment and fully online CorInfoMax experiments additionally demonstrate learning in deep residual architectures and biologically plausible recurrent networks with lateral connections. SBD thus offers both a practical broadcast-learning algorithm and a theoretical lens on biologically observed neuromodulatory plasticity.

Limitations. The strongest guarantees concern population optima under the stated assumptions; they do not establish convergence of the practical local no-propagation updates in finite networks trained on finite data. Although the depth experiments broaden the demonstrated applicability of SBD, substantial performance gaps to BP remain, and larger-scale evaluation is still needed. Improving local learning dynamics and developing data-adaptive score expansions are promising directions toward narrowing these gaps and enabling more effective and scalable broadcast learning. Relating the framework to measurements of neuromodulatory plasticity is a complementary direction.

AI USE STATEMENT

We utilized generative AI tools for grammatical editing, text refinement, LaTeX and reference formatting, and literature discovery. We also used these tools to generate and edit code implementations of our methods and numerical experiments, check consistency between the code and manuscript, and obtain feedback on experimental methodology and configurations.

We used LLM-based tools to check and refine the formulation of mathematical claims derived by the authors and to assist with writing and revising their proofs. The theoretical framework, derivation of the SBD learning algorithm, and essential proof arguments were developed by the authors. We reviewed the suggested mathematical refinements and corrections before incorporating them.

We have not used AI to propose or refine research hypotheses, provide critical ingredients for the proofs, assist with translation, or clean and reformat existing datasets. Qualitative and thematic data analysis is not applicable to this work.

We have reviewed all AI-assisted work including text and code. We also utilized AI-tools for literature discovery, which is followed by manual investigation and reporting of these references. We take responsibility for the final content of this work, including text, claims or code produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Experimental settings are described in the appendices. The supplementary repository contains the implemented SBD variants, selected baseline trainers, and the Poisson notebook; its README identifies the supplied experiments.

REFERENCES

Mohamed Akrout, Collin Wilson, Peter C. Humphreys, Timothy Lillicrap, and Douglas Tweed. Deep learning without weight transport. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 974–982, 2019.

Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E. Hinton, and Timothy Lillicrap. Assessing the Scalability of Biologically-Motivated Deep Learning Algorithms and Architectures. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 9368–9378, 2018.

Yoshua Bengio. How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation. *CoRR*, abs/1407.7906, 2014.

Lukas Biewald. Experiment tracking with Weights and Biases. Software available from wandb.com, 2020.

Blake Bordelon and Cengiz Pehlevan. The Influence of Learning Rule on Representation Dynamics in Wide Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2023.

Bariscan Bozkurt, Cengiz Pehlevan, and Alper T. Erdogan. Correlative Information Maximization: A Biologically Plausible Approach to Supervised Deep Neural Networks without Weight Symmetry. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, pages 34928–34941, 2023.

Bariscan Bozkurt, Ateş İsfendiyaroğlu, Cengiz Pehlevan, and Alper T. Erdogan. Correlative Information Maximization Based Biologically Plausible Neural Networks for Correlated Source Separation. In *International Conference on Learning Representations (ICLR)*, 2023.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

David G. Clark, L. F. Abbott, and SueYeon Chung. Credit Assignment Through Broadcasting a Global Error Vector. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2021.

Francis Crick. The recent excitement about neural networks. *Nature*, 337:129–132, 1989.

Mete Erdogan, Cengiz Pehlevan, and Alper Tunga Erdogan. Error Broadcast and Decorrelation as a Potential Artificial and Natural Learning Mechanism. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, 2025.

C. Févotte, N. Bertin, and J.-L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural Computation*, 21(3):793–830, 2009.

Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9:85, 2016.

Jerome H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: experimental support of neo-Hebbian three-factor learning rules. *Frontiers in Neural Circuits*, 12:53, 2018.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

S. Golkar, T. Tesileanu, Y. Bahroun, A. Sengupta, and D. Chklovskii. Constrained predictive coding as a biologically plausible model of the cortical hierarchy. *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pages 14155–14169, 2022.

Donghyeon Han and Hoi-jun Yoo. Efficient Convolutional Neural Network Training with Direct Feedback Alignment. *CoRR*, abs/1901.01986, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

Geoffrey E. Hinton. The Forward-Forward Algorithm: Some Preliminary Investigations. *CoRR*, abs/2212.13345, 2022.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 2015.

Fumitada Itakura and Shuzo Saito. Analysis synthesis telephony based on the maximum likelihood method. In *Proceedings of the 6th International Congress on Acoustics*, pages C–17–C–20. IEEE, 1968.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical Report, University of Toronto, 2009.

Łukasz Kuśmierz, Takuya Isomura, and Taro Toyoizumi. Learning with three factors: modulating Hebbian plasticity with errors. *Current Opinion in Neurobiology*, 46:170–177, 2017.

Julien Launay, Iacopo Poli, and Florent Krzakala. Principled Training of Neural Networks with Direct Feedback Alignment. *CoRR*, abs/1906.04554, 2019.

Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct Feedback Alignment Scales to Modern Deep Learning Tasks and Architectures. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Ya Le and Xuan Yang. Tiny ImageNet Visual Recognition Challenge. CS 231N course project report, Stanford University, 2015.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference Target Propagation. In *Machine Learning and Knowledge Discovery in Databases*, pages 498–515, 2015.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, 2016.

Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Arild Nøkland. Direct Feedback Alignment Provides Learning in Deep Neural Networks. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 1037–1045, 2016.

Arild Nøkland and Lars Hiller Eidnes. Training Neural Networks with Local Error Signals. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 4839–4850. PMLR, 2019.

Serdar Ozsoy, Shadi Hamdan, Sercan Arik, Deniz Yuret, and Alper Erdogan. Self-Supervised Learning with an Information Maximization Criterion. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pages 35240–35253, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 8024–8035, 2019.

Roman Pogodin, Yash Mehta, Timothy Lillicrap, and Peter E. Latham. Towards Biologically Plausible Convolutional Networks. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 13917–13928, 2021.

Shanshan Qin, Nayantara Mudur, and Cengiz Pehlevan. Contrastive Similarity Matching for Supervised Learning. *Neural Computation*, 33(5):1300–1328, 2021.

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2:79–87, 1999.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017.

Benjamin Scellier and Yoshua Bengio. Equivalence of equilibrium propagation and recurrent backpropagation. *Neural Computation*, 31(2):312–329, 2019.

Wolfram Schultz. Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80(1):1–27, 1998.

James C. R. Whittington and Rafal Bogacz. An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Computation*, 29(5):1229–1262, 2017.

James C. R. Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235–250, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016.

APPENDIX CONTENTS

A Deferred proofs for main-text results **14**
- A.1 Proof of theorem 1 . . . 14
- A.2 Proof of lemma 1 . . . 14
- A.3 Proof of theorem 2 . . . 14
- A.4 Cross-entropy sufficiency under a dense-feature assumption . . . 14
- A.5 Proof of proposition 1 . . . 14
- A.6 Proof of theorem 3 . . . 15

B When does a loss yield a conditionally mean-zero score? **15**
- B.1 Proof-of-concept demonstration: Poisson NLL regression . . . 18

C Cross entropy SBD algorithm **20**
- C.1 Background on MMSE estimation theory and EBD . . . 21
- C.2 Derivation of the SBD update rule . . . 22
- C.3 Extension to convolutional layers . . . 23
- C.4 Update complexity . . . 25

D Score vector expansion: general theory and cross entropy example **26**
- D.1 General formulation . . . 26
- D.2 Cross entropy instantiation . . . 27

E CIFAR-10 experimental setup **27**
- E.1 Score broadcast and decorrelation . . . 28
- E.2 Network architecture . . . 30
- E.3 Training hyperparameters . . . 31
- E.4 Accuracy curves . . . 32
- E.5 Multi-seed ablation of score-expansion modulators . . . 32
- E.6 Cosine alignment between SBD updates and backpropagation gradients . . . 34

F Tiny ImageNet experimental setup **35**
- F.1 Network architecture . . . 36
- F.2 Data pipeline and augmentation . . . 36
- F.3 Training hyperparameters . . . 37
- F.4 Accuracy curves . . . 37

G Correlative Information Maximization (CorInfoMax) Network Experiments **40**
- G.1 CorInfoMax background and score-broadcast learning . . . 40
- G.2 The 10-stage fully connected CorInfoMax experiment . . . 43
- G.3 Locally Connected CorInfoMax Experiments . . . 46

H SBD in a residual network **47**
- H.1 Residual architecture and local updates . . . 48
- H.2 Batch normalization . . . 49
- H.3 Contextual score expansion . . . 51
- H.4 Update scaling . . . 51
- H.5 Training and accuracy curves . . . 52

A Deferred proofs for main-text results

A.1 Proof of theorem 1

Since cross entropy is strictly proper, its population minimizer is the true conditional label distribution, i.e., $\mathbf{p}^\star(X) = \mathbf{q}(X) = \mathbb{E}[\mathbf{Y} \mid X]$. Therefore,

$$\mathbb{E}\big[\mathbf{p}^\star(X) - \mathbf{Y} \mid X\big] = \mathbf{p}^\star(X) - \mathbb{E}[\mathbf{Y} \mid X] = \mathbf{q}(X) - \mathbf{q}(X) = \mathbf{0}.$$

Hence the output score of the optimal cross entropy predictor is conditionally mean zero. □

This probability-space optimum is attained by finite softmax logits exactly when every conditional class probability is positive.

A.2 Proof of lemma 1

The integrability assumption $\mathbb{E}\big[\|\mathbf{U}\, g(X)^T\|\big] < \infty$ ensures that the (vector-valued) tower property applies. Because $g(X)$ is measurable with respect to X, it can be pulled out of the conditional expectation. Hence

$$\mathbb{E}\big[\mathbf{U} g(X)^T\big] = \mathbb{E}\Big[\mathbb{E}\big[\mathbf{U} g(X)^T \mid X\big]\Big] = \mathbb{E}\Big[\mathbb{E}[\mathbf{U} \mid X] g(X)^T\Big].$$

This is exactly the mechanism by which conditional mean-zero leads to orthogonality with arbitrary functions of the input. □

A.3 Proof of theorem 2

By Theorem 1, $\mathbf{U} = \mathbf{p}^\star(X) - \mathbf{Y}$ satisfies $\mathbb{E}[\mathbf{U} \mid X] = \mathbf{0}$. Since $\mathbf{p}^\star(X) \in \Delta_D$ and $\mathbf{Y} \in \{0,1\}^D$, the score vector $\mathbf{U}$ is uniformly bounded, so any measurable g with $\mathbb{E}\|g(X)\| < \infty$ automatically yields an integrable product $\mathbf{U}\, g(X)^T$. Applying Lemma 1 to this choice of $\mathbf{U}$ immediately gives

$$\mathbb{E}\big[(\mathbf{p}^\star(X) - \mathbf{Y}) g(X)^T\big] = \mathbf{0}$$

for every such $g(X)$. Substituting $g(X) = g^{(k)}(\mathbf{h}^{(k)}(X))$ yields the layerwise version used by the decorrelation framework. □

A.4 Cross-entropy sufficiency under a dense-feature assumption

Theorem 2 shows that score–feature orthogonality is *necessary* at the population cross entropy optimum. The converse, that exact orthogonality identifies the optimum, requires the feature family used in the decorrelation to be sufficiently rich. Let $\mathbf{p}(X)$ denote a *candidate* predictor (not assumed to be the population minimizer), and define $m_{\mathbf{p}}(X) := \mathbb{E}[\mathbf{p}(X) - \mathbf{Y} \mid X] = \mathbf{p}(X) - \mathbf{q}(X)$, the gap between the candidate predictor and the true conditional class probabilities. The decorrelation condition $\mathbb{E}[(\mathbf{p}(X) - \mathbf{Y})\mathbf{z}(X)^T] = 0$ is, by the tower property, exactly the orthogonality of $m_{\mathbf{p}}$ against the feature $\mathbf{z}$ in $L^2(P_X)$. If the hidden-layer features (e.g., first-layer activations in the wide-network limit) have linear span dense in $L^2(P_X)$, then enforcing the decorrelation condition against *all* such features forces $m_{\mathbf{p}}$ to be orthogonal to a dense subspace. Since $m_{\mathbf{p}} \in L^2(P_X)$, this forces $m_{\mathbf{p}}(X) = 0$ almost surely, which gives $\mathbf{p}(X) = \mathbf{q}(X) = \mathbf{p}^\star(X)$, i.e., the candidate predictor coincides with the population minimizer of Theorem 1. The same dense-feature reasoning was used in EBD (Erdogan et al., 2025, Appendix B.2) for MSE; under SBD it operates identically with the cross entropy score. The finite-width algorithm should be viewed as an approximation to this idealized condition.

A.5 Proof of proposition 1

Proof. At a stationary point, by definition, the gradient of the population risk with respect to the last-layer weights must vanish. Since

$$\nabla_{\mathbf{W}^{(L)}} \mathcal{R} = \mathbb{E}\big[\boldsymbol{\delta}\, \mathbf{h}^{(L-1)T}\big],$$

setting the gradient to zero gives the claimed orthogonality relation immediately. □

A.6 PROOF OF THEOREM 3

The proof is the same tower-property argument used in the cross entropy case. Since $g(X)$ is measurable with respect to X,

$$\mathbb{E}\big[\boldsymbol{\delta}^\star g(X)^T\big] = \mathbb{E}\Big[\mathbb{E}\big[\boldsymbol{\delta}^\star g(X)^T \mid X\big]\Big] = \mathbb{E}\Big[\mathbb{E}[\boldsymbol{\delta}^\star \mid X] g(X)^T\Big].$$

Using the hypothesis $\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}$ gives

$$\mathbb{E}\big[\boldsymbol{\delta}^\star g(X)^T\big] = \mathbb{E}[\mathbf{0}\, g(X)^T] = \mathbf{0}.$$

Therefore the optimal score is orthogonal to every measurable function of the input. □

B WHEN DOES A LOSS YIELD A CONDITIONALLY MEAN-ZERO SCORE?

This appendix section provides the full conditional-risk characterization summarized in Section 5.

Theorem 3 shows that the generalized SBD framework has its full layerwise orthogonality property whenever the population-optimal score satisfies $\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}$. The key question is therefore: when does a loss produce this conditional mean-zero score? In order to perform this characterization, we look at the loss structure.

For a fixed input value x, define the conditional risk

$$C_x(\mathbf{a}) = \mathbb{E}\big[\mathcal{L}(\mathbf{Y}, \mathbf{a}) \mid X = x\big]. \tag{2}$$

Here $\mathbf{a}$ denotes the output parameter with respect to which the SBD score is taken. Throughout the theorem below, $\mathbf{a}$ lies in an open subset $\mathcal{A} \subseteq \mathbb{R}^m$. For MSE loss, $\mathbf{a}$ is directly the Euclidean estimate. For logit-based classification or natural-parameter likelihood models, $\mathbf{a}$ is the corresponding unconstrained output parameter. If a loss is naturally defined on a constrained prediction variable, such as a probability vector $\mathbf{p} \in \Delta_D$, we treat it here by composing it with an unconstrained link $\mathbf{p} = \psi(\mathbf{a})$ and applying the theorem to

$$\mathcal{L}(\mathbf{Y}, \mathbf{a}) = \ell(\mathbf{Y}, \psi(\mathbf{a})).$$

The direct constrained formulation is not needed for the SBD results in this paper and is left outside the main theorem.

We note that the population risk is related to the conditional risk in Eq. (2) simply through

$$\mathcal{R}(\mathbf{a}) = \mathbb{E}\big[\mathcal{L}(\mathbf{Y}, \mathbf{a}(X))\big] = \mathbb{E}_X\big[C_X(\mathbf{a}(X))\big].$$

A measurable predictor that minimizes C_x for almost every x therefore minimizes population risk, provided the relevant risks are finite.

Theorem 5 (Conditional-risk characterization in an unconstrained parameterization) *Let $\mathcal{A} \subseteq \mathbb{R}^m$ be open, and let $\mathcal{L}(\mathbf{y}, \mathbf{a})$ be differentiable in $\mathbf{a} \in \mathcal{A}$. For almost every x, assume that the conditional risk $C_x(\mathbf{a})$ has an interior minimizer $\mathbf{a}^\star(x) \in \mathcal{A}$, and that differentiation may be interchanged with conditional expectation. Then the optimal score*

$$\boldsymbol{\delta}^\star = \nabla_{\mathbf{a}} \mathcal{L}(\mathbf{Y}, \mathbf{a}^\star(X))$$

satisfies

$$\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}.$$

If, in addition, $\mathcal{A}$ is convex and $C_x(\mathbf{a})$ is convex in $\mathbf{a}$ for almost every x, then the converse also holds: any measurable predictor $\tilde{\mathbf{a}}(X)$ satisfying

$$\mathbb{E}\big[\nabla_{\mathbf{a}} \mathcal{L}(\mathbf{Y}, \tilde{\mathbf{a}}(X)) \mid X\big] = \mathbf{0}$$

minimizes $C_x(\mathbf{a})$ for almost every x; if $C_x(\mathbf{a})$ is strictly convex, this minimizer is unique almost surely.

Proof. Forward part of the proof simply relies on the first order condition for optimality, and being able to exchange differentiation and expectation under the presumed regularity assumption. For each fixed input value x, differentiation of the conditional risk gives

$$\nabla_{\mathbf{a}} C_x(\mathbf{a}) = \nabla_{\mathbf{a}} \mathbb{E}\big[\mathcal{L}(\mathbf{Y}, \mathbf{a}) \mid X = x\big].$$

If $\mathbf{a}^\star(x)$ is an interior minimizer of the differentiable function C_x, then its first-order optimality condition is

$$\nabla_{\mathbf{a}} C_x(\mathbf{a}^\star(x)) = \mathbf{0}.$$

Exchanging differentiation with expectation yields

$$\mathbb{E}\big[\nabla_{\mathbf{a}} \mathcal{L}(\mathbf{Y}, \mathbf{a}^\star(x)) \mid X = x\big] = \mathbf{0}$$

for almost every x, which is exactly $\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}$.

For the reverse direction, suppose C_x is differentiable and convex and a predictor $\tilde{\mathbf{a}}(x)$ satisfies

$$\mathbb{E}\big[\nabla_{\mathbf{a}} \mathcal{L}(\mathbf{Y}, \tilde{\mathbf{a}}(X)) \mid X\big] = \mathbf{0},$$

We show that $\tilde{\mathbf{a}}(x)$ minimizes C_x for almost every x.

First, we translate the score condition into a gradient condition on C_x. Applying the gradient–expectation interchange used in the forward direction, but now evaluated at $\tilde{\mathbf{a}}(X)$ rather than at $\mathbf{a}^\star(X)$, gives

$$\nabla_{\mathbf{a}} C_x(\tilde{\mathbf{a}}(x)) = \mathbb{E}\big[\nabla_{\mathbf{a}} \mathcal{L}(\mathbf{Y}, \tilde{\mathbf{a}}(x)) \mid X = x\big].$$

The right-hand side is $\mathbf{0}$ for almost every x by assumption. Therefore

$$\nabla_{\mathbf{a}} C_x(\tilde{\mathbf{a}}(x)) = \mathbf{0} \quad \text{for almost every } x.$$

As the next step, we use the fact that a zero gradient of a convex function characterizes a global minimizer. Fix any x for which C_x is differentiable, convex, and satisfies $\nabla_{\mathbf{a}} C_x(\tilde{\mathbf{a}}(x)) = \mathbf{0}$. By the first-order characterization of convexity, for every $\mathbf{a} \in \mathcal{A}$,

$$C_x(\mathbf{a}) \geq C_x(\tilde{\mathbf{a}}(x)) + \nabla_{\mathbf{a}} C_x(\tilde{\mathbf{a}}(x))^T(\mathbf{a} - \tilde{\mathbf{a}}(x)) = C_x(\tilde{\mathbf{a}}(x)) + \mathbf{0}^T(\mathbf{a} - \tilde{\mathbf{a}}(x)) = C_x(\tilde{\mathbf{a}}(x)).$$

Hence $\tilde{\mathbf{a}}(x)$ is a global minimizer of C_x. Since this holds for almost every x, the predictor $\tilde{\mathbf{a}}$ minimizes C_x pointwise almost surely.

Finally, in case of strict convexity of C_x, the inequality above is strict whenever $\mathbf{a} \neq \tilde{\mathbf{a}}(x)$:

$$C_x(\mathbf{a}) > C_x(\tilde{\mathbf{a}}(x)) \quad \text{for all } \mathbf{a} \in \mathcal{A} \setminus \{\tilde{\mathbf{a}}(x)\}.$$

Therefore $\tilde{\mathbf{a}}(x)$ is the *unique* minimizer of C_x, and the converse predictor is unique almost surely. □

The theorem clarifies why this condition is stronger than ordinary stationarity of a parameterized model $\mathbf{a}_\theta(X)$. When optimization is carried out only over the parameter vector θ, one generally obtains the weaker feature-level orthogonality of Proposition 1. The stronger identity $\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}$ appears when the optimal prediction value is characterized directly from the conditional risk for each input.

We apply the theorem to Bregman divergences (covering MSE), proper scoring rules through unconstrained links (covering cross-entropy), and exponential-family negative log-likelihoods.

Bregman losses. Bregman divergences (Bregman, 1967) form a broad family of losses generated by a strictly convex potential function ϕ. Important members include MSE (from $\phi(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|_2^2$), the Itakura–Saito divergence (Itakura and Saito, 1968; Févotte et al., 2009), and the generalized KL divergence. We show that for any such loss, the population-optimal predictor is a conditional expectation, and the score is therefore conditionally mean zero.

Let $\Omega \subseteq \mathbb{R}^m$ be open and convex, and let $\phi : \Omega \to \mathbb{R}$ be twice differentiable with positive definite Hessian $\nabla^2\phi$. Let $\tau : \mathcal{Y} \to \Omega$ be a target encoding. Assume finite conditional first moments and Bregman risks, with $\boldsymbol{\mu}_x = \mathbb{E}[\tau(\mathbf{Y}) \mid X = x] \in \Omega$. The associated Bregman divergence and loss are

$$D_\phi(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u}) - \phi(\mathbf{v}) - \nabla\phi(\mathbf{v})^T(\mathbf{u} - \mathbf{v}), \qquad \mathcal{L}_\phi(\mathbf{y}, \mathbf{a}) = D_\phi(\tau(\mathbf{y}), \mathbf{a}).$$

Differentiating with respect to the second argument gives

$$\nabla_{\mathbf{a}} \mathcal{L}_\phi(\mathbf{y}, \mathbf{a}) = \nabla^2\phi(\mathbf{a})\big(\mathbf{a} - \tau(\mathbf{y})\big).$$

Taking the conditional expectation over $\mathbf{Y} \mid X = x$ and using linearity yields the gradient of the conditional risk:

$$\nabla_{\mathbf{a}} C_x(\mathbf{a}) = \nabla^2 \phi(\mathbf{a}) \Big(\mathbf{a} - \mathbb{E}[\tau(\mathbf{Y}) \mid X = x] \Big).$$

Global optimality follows from the Bregman decomposition

$$C_x(\mathbf{a}) = C_x(\boldsymbol{\mu}_x) + D_\phi(\boldsymbol{\mu}_x, \mathbf{a}).$$

Strict convexity of ϕ makes the second term nonnegative, with equality only at $\mathbf{a} = \boldsymbol{\mu}_x$. Thus the unique minimizer is

$$\mathbf{a}^\star(X) = \mathbb{E}[\tau(\mathbf{Y}) \mid X].$$

This is precisely the form required by Theorem 5, so the optimal score $\boldsymbol{\delta}^\star = \nabla_{\mathbf{a}} \mathcal{L}_\phi(\mathbf{Y}, \mathbf{a}^\star(X))$ satisfies $\mathbb{E}[\boldsymbol{\delta}^\star \mid X] = \mathbf{0}$. The MSE case $\phi(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|_2^2$, $\tau(\mathbf{Y}) = \mathbf{Y}$ gives $\mathbf{a}^\star(X) = \mathbb{E}[\mathbf{Y} \mid X]$ and $\boldsymbol{\delta}^\star = \mathbf{a}^\star(X) - \mathbf{Y}$, recovering the MMSE estimation-error orthogonality used by EBD as a single Bregman instance.

Proper probability losses through unconstrained links. Probabilistic classification uses losses defined on the probability simplex: cross-entropy, Brier score, log loss, and so on. These losses share a strong optimality property (Gneiting and Raftery, 2007): their conditional risks are minimized at the true conditional class-probability vector. The complication is that the simplex Δ_D is a constrained domain, so Theorem 5 does not apply directly. The standard fix is to compose the simplex-domain loss with an unconstrained link function (e.g., softmax over logits), which moves the problem into the open Euclidean setting where Theorem 5 applies.

Let $\ell(\mathbf{y}, \mathbf{p})$ be a differentiable strictly proper loss defined on the interior of the probability simplex, and let

$$\mathbf{q}(X) = \mathbb{E}[\mathbf{Y} \mid X]$$

denote the true conditional class-probability vector, assumed to lie in the interior of the simplex almost surely. Strict propriety means that, for each fixed x, the simplex-domain conditional risk

$$\mathbf{p} \mapsto \mathbb{E}\big[\ell(\mathbf{Y}, \mathbf{p}) \mid X = x\big]$$

is uniquely minimized at $\mathbf{p} = \mathbf{q}(x)$.

To bring this into the unconstrained setting required by Theorem 5, we reparameterize through a differentiable link $\psi : \mathbb{R}^m \to \mathrm{int}(\Delta_D)$ that maps an unconstrained vector $\mathbf{a}$ to a probability vector $\mathbf{p} = \psi(\mathbf{a})$, and define the unconstrained loss

$$\mathcal{L}(\mathbf{y}, \mathbf{a}) = \ell(\mathbf{y}, \psi(\mathbf{a})).$$

The SBD score is then taken with respect to the unconstrained variable:

$$\boldsymbol{\delta} = \nabla_{\mathbf{a}} \mathcal{L}(\mathbf{y}, \mathbf{a}).$$

Suppose that for almost every x, the unconstrained conditional risk

$$C_x(\mathbf{a}) = \mathbb{E}\big[\ell(\mathbf{Y}, \psi(\mathbf{a})) \mid X = x\big]$$

has an interior minimizer $\mathbf{a}^\star(x)$ satisfying $\psi(\mathbf{a}^\star(x)) = \mathbf{q}(x)$, and that the gradient–expectation interchange holds. Then by Theorem 5,

$$\mathbb{E}\big[\nabla_{\mathbf{a}} \ell(\mathbf{Y}, \psi(\mathbf{a}^\star(X))) \mid X\big] = \mathbf{0},$$

which is exactly the conditional-mean-zero score property used by SBD.

Softmax cross-entropy. The canonical instance is $\psi = \mathrm{softmax}$, $\ell(\mathbf{y}, \mathbf{p}) = -\sum_{d=1}^{D} y_d \log p_d$. A direct computation gives

$$\mathbf{p} = \mathrm{softmax}(\mathbf{a}), \qquad \nabla_{\mathbf{a}} \mathcal{L}_{\mathrm{CE}}(\mathbf{y}, \mathbf{a}) = \mathbf{p} - \mathbf{y}.$$

At the population optimum, $\mathbf{p}^\star(X) = \psi(\mathbf{a}^\star(X)) = \mathbf{q}(X)$, so

$$\mathbb{E}\big[\mathbf{p}^\star(X) - \mathbf{Y} \mid X\big] = \mathbf{p}^\star(X) - \mathbf{q}(X) = \mathbf{0}.$$

This is the cross-entropy score $\mathbf{p} - \mathbf{y}$ used throughout the main text, recovered as the unconstrained-logit specialization of the proper-loss construction.

Negative log-likelihood losses. Maximum-likelihood estimation in a parametric family yields a third source of differentiable losses. We show that whenever the parameter space is open and the conditional risk has an interior minimizer, the score is conditionally mean zero. Furthermore, for exponential families, this conditional-mean-zero property reduces to the classical moment-matching identity.

Let $\{p_\eta : \eta \in \mathcal{H}\}$ be a differentiable parametric family of densities with open parameter space $\mathcal{H} \subseteq \mathbb{R}^m$, and define the negative log-likelihood loss

$$\mathcal{L}_{\mathrm{NLL}}(\mathbf{y}, \eta) = -\log p_\eta(\mathbf{y}).$$

The corresponding conditional risk is the conditional cross-entropy from the true distribution to p_η,

$$C_x(\eta) = \mathbb{E}\big[-\log p_\eta(\mathbf{Y}) \mid X = x\big].$$

Suppose C_x has an interior minimizer $\eta^\star(x) \in \mathcal{H}$ for almost every x, and the gradient–expectation interchange holds. Theorem 5 then gives

$$\mathbb{E}\big[-\nabla_\eta \log p_{\eta^\star(X)}(\mathbf{Y}) \mid X\big] = \mathbf{0}.$$

The NLL score is the negative of the statistical likelihood score $\nabla_\eta \log p_\eta(\mathbf{Y})$; both have zero conditional mean here.

Exponential families: moment matching. For an exponential family in canonical form,

$$p_\eta(\mathbf{y}) = h(\mathbf{y}) \exp\big(\eta^T T(\mathbf{y}) - A(\eta)\big),$$

the score has a particularly clean form: $\nabla_\eta \log p_\eta(\mathbf{y}) = T(\mathbf{y}) - \nabla A(\eta)$, so

$$\boldsymbol{\delta}_{\mathrm{NLL}} = \nabla A(\eta) - T(\mathbf{Y}).$$

The conditional-mean-zero score condition $\mathbb{E}[\boldsymbol{\delta}^\star_{\mathrm{NLL}} \mid X] = \mathbf{0}$ then reads

$$\nabla A(\eta^\star(X)) = \mathbb{E}[T(\mathbf{Y}) \mid X],$$

which is exactly the classical moment-matching condition for maximum-likelihood estimation in exponential families: the model expected sufficient statistic equals the conditional expected sufficient statistic. Important instances include Gaussian NLL (with $T(\mathbf{Y}) = \mathbf{Y}$, recovering the conditional-mean predictor), Bernoulli logistic loss, Poisson NLL, and multinomial softmax cross-entropy when written in natural-parameter coordinates.

B.1 Proof-of-concept demonstration: Poisson NLL regression

The main paper's experiments target cross-entropy classification, which together with the MSE setting of Erdogan et al. (2025) covers two of the loss families admitted by SBD. To illustrate the framework on an exponential-family negative log-likelihood outside these two, we run a small synthetic Poisson regression experiment within the scope of Theorem 5 and Appendix B. We compare BP and SBD through test NLL, binned residual calibration, and score–activation correlations, using the known population-optimal predictor as an oracle reference.

B.1.1 Data-generating process.

We sample $X \sim \mathcal{N}(\mathbf{0}, I_8)$ and define the ground-truth log-rate

$f^\star(x) = 1.0 + 0.4\sin(x_1)\cos(x_2) + 0.3\,x_3x_4 - 0.15(x_5^2 - 1) + 0.10(x_6 + x_7) + 0.10\tanh(x_8)$,

as a synthetic regression function in the style of Friedman 1 (Friedman, 1991), so that $Y \mid X \sim \mathrm{Poisson}\big(\exp f^\star(X)\big)$. We generate $50{,}000$ training and $10{,}000$ test samples, with $f^\star$ clipped to $[-1.5, 3.5]$ before sampling Y to bound the generating rates. Hereafter, $f^\star$ denotes this clipped log-rate. Because it is known explicitly, we can estimate the Bayes NLL $\mathcal{R}^\star = \mathbb{E}[\exp f^\star(X) - Y\, f^\star(X)]$ and the oracle's finite-sample calibration diagnostic (defined below).

B.1.2 Loss and score.

The Poisson NLL with log-rate parameterization is $\mathcal{L}_{\mathrm{NLL}}(y, a) = \exp(a) - y\,a$ (up to a y-only constant), giving the SBD score

$$\boldsymbol{\delta}(y, a) = \nabla_a \mathcal{L}_{\mathrm{NLL}}(y, a) = \exp(a) - y.$$

This is the canonical exponential-family score $\nabla A(\eta) - T(Y)$ with $T(Y) = Y$ and $A(\eta) = \exp(\eta)$, matching the structure of Appendix B. The conditional-mean-zero property at the optimum reduces to the moment-matching identity $\exp(a^\star(X)) = \mathbb{E}[Y \mid X]$.

B.1.3 Derivation of the oracle Bayes NLL.

The Poisson NLL on a sample (X, Y) in log-rate parameterization is, up to a Y-only constant,

$$\mathcal{L}_{\mathrm{NLL}}(Y, a) = \exp(a) - Y\, a.$$

The conditional risk for a fixed input x is

$$C_x(a) = \mathbb{E}\big[\mathcal{L}_{\mathrm{NLL}}(Y, a) \mid X = x\big] = \exp(a) - \mathbb{E}[Y \mid X = x]\, a.$$

For the synthetic data-generating process $Y \mid X \sim \mathrm{Poisson}\big(\exp f^\star(X)\big)$, the conditional mean is $\mathbb{E}[Y \mid X = x] = \exp f^\star(x)$. Differentiating C_x in a and setting the gradient to zero gives $\exp(a^\star(x)) = \exp f^\star(x)$, so the population-optimal predictor is $a^\star(x) = f^\star(x)$. Substituting back yields the conditional risk at the optimum

$$C_x(a^\star(x)) = \exp f^\star(x) \cdot \big(1 - f^\star(x)\big).$$

The oracle Bayes NLL is the marginal expectation of this conditional risk:

$$\mathcal{R}^\star = \mathbb{E}_X\big[C_X(a^\star(X))\big] = \mathbb{E}\big[\exp f^\star(X) - Y\, f^\star(X)\big],$$

where the second equality uses the tower property and $\mathbb{E}[Y \mid X] = \exp f^\star(X)$. Because $f^\star$ is known, we estimate $\mathcal{R}^\star$ by Monte Carlo on the test set, $\widehat{\mathcal{R}}^\star = N_{\mathrm{test}}^{-1} \sum_{n=1}^{N_{\mathrm{test}}} \big[\exp f^\star(X_n) - Y_n\, f^\star(X_n)\big]$, which converges to $\mathcal{R}^\star$ at rate $N_{\mathrm{test}}^{-1/2}$. For the test set of $10{,}000$ samples used here, this yields $\widehat{\mathcal{R}}^\star \approx -0.5595$. This empirical oracle value estimates the population Bayes risk; it is not a lower bound for every finite-sample test NLL.

B.1.4 Network and methods.

We use a three-layer MLP with widths $8 \to 128 \to 64 \to 1$, ReLU activations, bias terms, and Kaiming–He initialization (He et al., 2015). The network output is the scalar log-rate $a(x)$. Two methods are compared:

- **BP:** backpropagation of the exact Poisson NLL gradient.
- **SBD:** the rank-1 score broadcast of Section 4 (no score-vector expansion). The output layer uses the standard NLL gradient; the two hidden layers use the local SBD update with broadcast cross-correlation $\widehat{\mathbf{R}}^{(k)}_{\mathrm{Score}}$ projecting $\boldsymbol{\delta} = \exp(a) - y$ to each layer.

Both methods use Adam with $\eta_0 = 3 \times 10^{-3}$, 0.99 per-epoch learning-rate decay, batch size 64, and 200 epochs over five runs. Weight decay is 5×10^{-4} for BP and 10^{-5} for SBD. SBD uses $\lambda = 0.99999$ for the broadcast exponential moving average (EMA), initialized with independent $\mathcal{N}(0, 0.01^2)$ entries.

B.1.5 Evaluation metrics.

We track test NLL and binned residual calibration per epoch, and evaluate score–activation correlations after training.

- *Test Poisson NLL.* The test-average loss $\widehat{\mathcal{R}}$, compared with the empirical oracle $\widehat{\mathcal{R}}^\star$ on the same test set.
- *Binned residual calibration (CMZ in the figures).* We bin the test set into $K = 20$ quantile bins of the predicted log-rate $\hat{a}(X)$, compute $|\widehat{\mathbb{E}}[\boldsymbol{\delta} \mid \mathrm{bin}_k]|$ within each bin, and average the absolute values. This tests residual calibration with respect to binned predictions, a coarser condition than $\mathbb{E}[\boldsymbol{\delta} \mid X] = 0$. Replacing $\hat{a}$ with $f^\star$ in both the score and binning gives an oracle reference of 0.0628; finite-sample fluctuations make this reference nonzero, but it is not a lower bound on the diagnostic.
- *Score–activation correlations.* Average absolute Pearson correlation between $\boldsymbol{\delta}$ and each hidden-layer activation, evaluated on the test set for one trained network per method. These are empirical diagnostics of the orthogonality predicted at the population optimum by Theorem 3.

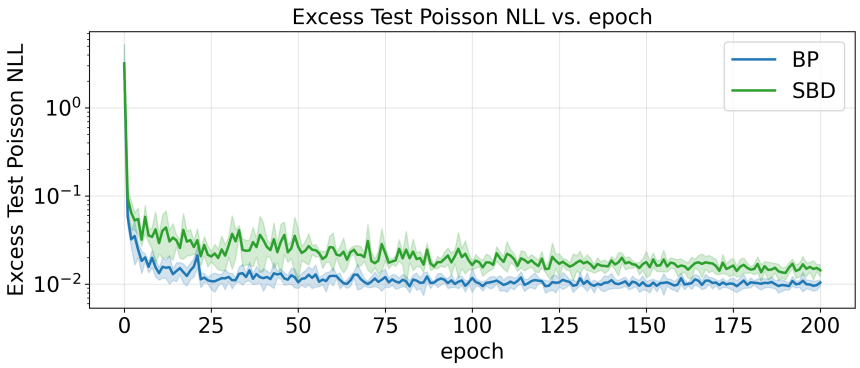


Figure 2: Poisson proof-of-concept: test excess negative log-likelihood relative to the empirical oracle, $\widehat{\mathcal{R}} - \widehat{\mathcal{R}}^\star$, on a logarithmic scale. Both BP and SBD converge close to the oracle, with SBD plateauing slightly above BP. Shading denotes one standard deviation across five runs.

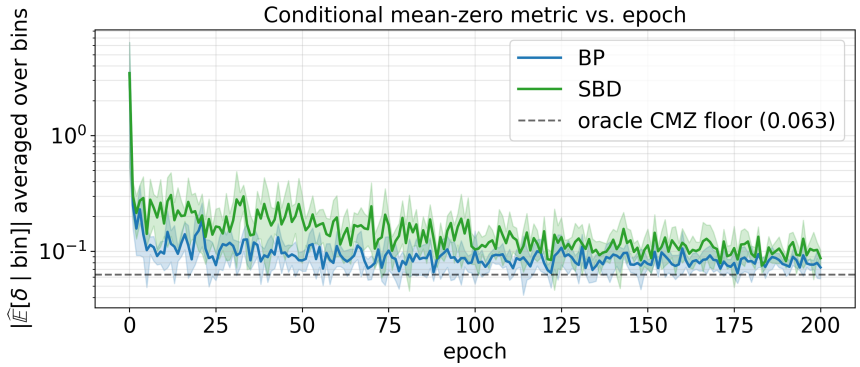


Figure 3: Poisson proof-of-concept: binned residual calibration versus epoch on a logarithmic scale. Shading denotes one standard deviation across five runs. The dashed line, labeled "oracle CMZ floor," is the oracle's finite-sample diagnostic value, not a lower bound.

Results. Both methods converge to near-oracle test Poisson NLL: BP attains a final test NLL of -0.5491 ± 0.0008 and SBD -0.5452 ± 0.0023, compared with $\widehat{\mathcal{R}}^\star = -0.5595$. Reported uncertainties and shaded bands are standard deviations across five runs, computed with divisor $n = 5$ as in the notebook. Figure 2 shows the test NLL excess over the oracle ($\widehat{\mathcal{R}} - \widehat{\mathcal{R}}^\star$) on a logarithmic scale. Both BP and SBD descend by roughly two and a half orders of magnitude during training and plateau at small residual gaps, approximately 1×10^{-2} for BP and 1.5×10^{-2} for SBD.

Figure 3 shows binned residual calibration versus epoch. Both methods reduce the diagnostic by more than an order of magnitude: BP reaches 0.0726 ± 0.0146 and SBD 0.0872 ± 0.0156, compared with the oracle reference of 0.0628. Its decrease accompanies the decline in test NLL. The post-training score–activation correlations range from 0.006 to 0.019 across the two hidden layers and methods.

Together, near-oracle test NLL, improved residual calibration, and small score–activation correlations support the use of SBD on this Poisson task. These finite-sample diagnostics are consistent with the population theory, but do not establish full conditional mean zero or attainment of the population optimum.

C Cross entropy SBD algorithm

This appendix collects the cross entropy SBD procedure used in Section 3 into a single self-contained reference. Table 3 lists the per-minibatch operations performed at every hidden layer k, where $\mathbf{h}^{(k)} \in \mathbb{R}^{N^{(k)}}$ is the layer-k activation, $\mathbf{a} \in \mathbb{R}^{D_{\text{out}}}$ are the output logits, $\mathbf{p} = \text{softmax}(\mathbf{a})$, and

$\boldsymbol{\delta}_{\mathrm{CE}} = \mathbf{p} - \mathbf{y}$ is the cross entropy output score. The layerwise correlation estimate $\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)} \in \mathbb{R}^{N^{(k)} \times D_{\mathrm{out}}}$ tracks the running cross-correlation between the hidden feature map $g^{(k)}(\mathbf{h}^{(k)})$ and the score, and supplies the broadcast modulator $\mathbf{q}_{\mathrm{CE}}^{(k)}$ used in the local three-factor weight update. The output layer is trained with the standard cross entropy gradient.

Table 3: Cross entropy SBD algorithm.

Step	Operation
1	Perform a forward pass to compute the hidden activations $\mathbf{h}^{(k)}$, the logits $\mathbf{a}$, the probabilities $\mathbf{p} = \mathrm{softmax}(\mathbf{a})$, and the score $\boldsymbol{\delta}_{\mathrm{CE}} = \mathbf{p} - \mathbf{y}$.
2	Form the batch score matrix $\boldsymbol{\Delta}_{\mathrm{CE}}[m] = \mathbf{P}[m] - \mathbf{Y}[m]$ and update the layerwise correlation estimate $\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m] = \lambda\, \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m-1] + \frac{1-\lambda}{B}\, \mathbf{G}^{(k)}[m]\, \boldsymbol{\Delta}_{\mathrm{CE}}[m]^T$.
3	For $n = mB+1, \ldots, (m+1)B$, project the broadcast score to hidden layer k by computing $\mathbf{q}_{\mathrm{CE}}^{(k)}[n] = \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]\, (\mathbf{p}[n] - \mathbf{y}[n])$.
4	For $B = 1$, update the hidden-layer parameters with the local three-factor rule $\Delta W_{ij}^{(k)}[n] = \zeta\, g_i'^{(k)}\big(h_i^{(k)}[n]\big)\, f'^{(k)}\big(u_i^{(k)}[n]\big)\, q_{\mathrm{CE},i}^{(k)}[n]\, h_j^{(k-1)}[n]$, and update the biases analogously; for $B > 1$, use Eq. (4).
5	Update the output layer with the standard cross entropy gradient.

C.1 Background on MMSE estimation theory and EBD

Error Broadcast and Decorrelation (EBD) begins from a standard property of nonlinear minimum mean square error (MMSE) estimation (Erdogan et al., 2025). For random vectors $\mathbf{X}$ and $\mathbf{Y}$, the optimal MMSE estimator is the conditional mean

$$\widehat{\mathbf{y}}^{\star}(\mathbf{X}) = \mathbb{E}[\mathbf{Y} \mid \mathbf{X}].$$

Its estimation error $\boldsymbol{\epsilon}^{\star} = \mathbf{Y} - \widehat{\mathbf{y}}^{\star}(\mathbf{X})$ satisfies the stochastic orthogonality condition: it is orthogonal to every measurable function $\mathbf{g}(\mathbf{X})$ of the input for which the expectation exists,

$$\mathbb{E}[\boldsymbol{\epsilon}^{\star} \mid \mathbf{X}] = \mathbf{0} \quad \Longrightarrow \quad \mathbb{E}\big[\mathbf{g}(\mathbf{X})\boldsymbol{\epsilon}^{\star T}\big] = \mathbf{0}. \tag{3}$$

EBD further uses the sufficiency of this condition for nonlinear MMSE estimation: when the available functions are sufficiently rich, enforcing their orthogonality with the estimation error identifies the conditional-mean estimator (Erdogan et al., 2025).

EBD applies this principle to a neural network trained with MSE. Writing its output error as $\boldsymbol{\epsilon} = \mathbf{h}^{(L)} - \mathbf{y}$, the hidden-layer activations and their nonlinear transforms $\mathbf{g}^{(k)}(\mathbf{h}^{(k)})$ provide measurable functions of the network input. At each hidden layer, EBD therefore seeks to enforce zero cross-correlation between the transformed layer activations and the output error:

$$\mathbf{R}_{\mathbf{g}\boldsymbol{\epsilon}}^{(k)} = \mathbb{E}\Big[\mathbf{g}^{(k)}(\mathbf{h}^{(k)})\boldsymbol{\epsilon}^T\Big], \qquad \mathcal{J}_{\mathrm{EBD}}^{(k)} = \frac{1}{2}\left\|\mathbf{R}_{\mathbf{g}\boldsymbol{\epsilon}}^{(k)}\right\|_F^2.$$

For a minibatch of size B, with $\mathbf{G}^{(k)}[m] \in \mathbb{R}^{N^{(k)} \times B}$ and $\mathbf{E}[m] \in \mathbb{R}^{D_{\mathrm{out}} \times B}$, this correlation is estimated autoregressively with forgetting factor λ:

$$\widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\epsilon}}^{(k)}[m] = \lambda \widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\epsilon}}^{(k)}[m-1] + \frac{1-\lambda}{B}\mathbf{G}^{(k)}[m]\mathbf{E}[m]^T, \qquad \widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\epsilon}}^{(k)}[m] \in \mathbb{R}^{N^{(k)} \times D_{\mathrm{out}}}.$$

The derivative of the layer-specific loss with respect to a layer parameter decomposes into two gradient components. The first contains derivatives of the transformed layer activations; the second contains derivatives of the output error. The second component requires signals propagated backward from the output layer, whereas the first depends only on the layer activations and the broadcast output error. EBD retains the first component to obtain its localized update. For a single sample, it defines

$$\mathbf{q}^{(k)} = \widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\epsilon}}^{(k)}\boldsymbol{\epsilon}, \qquad \Delta W_{ij}^{(k)} = \zeta\, g_i'^{(k)}(h_i^{(k)}) f'^{(k)}(u_i^{(k)}) q_i^{(k)} h_j^{(k-1)}.$$

Here $\mathbf{q}^{(k)}$ is the projection of the output error onto the layer activations through the estimated cross-correlation matrix. The update has the three-factor form emphasized in EBD: the presynaptic activity

$h_j^{(k-1)}$, the postsynaptic factor $g_i'^{(k)} f'^{(k)}$, and the projected global error $q_i^{(k)}$ as the modulatory factor. EBD is tied to the MSE output error and its MMSE orthogonality property; SBD retains this broadcast-and-decorrelate construction while replacing the output error with the appropriate loss score. The derivation below specializes that construction to cross entropy.

C.2 Derivation of the SBD update rule

This subsection derives the practical hidden-layer SBD update directly from the minibatch correlation objective. For each sample n, the layer-k preactivation and activation satisfy

$$\mathbf{u}^{(k)}[n] = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}[n] + \mathbf{b}^{(k)}, \qquad \mathbf{h}^{(k)}[n] = f^{(k)}\Big(\mathbf{u}^{(k)}[n]\Big),$$

and $g^{(k)}$ denotes the elementwise nonlinear transform used in the score–activation correlation.

The batch probability and label matrices are

$$\mathbf{P}[m] = \big[\mathbf{p}[mB+1], \ldots, \mathbf{p}[mB+B]\big], \qquad \mathbf{Y}[m] = \big[\mathbf{y}[mB+1], \ldots, \mathbf{y}[mB+B]\big]$$

with $\mathbf{\Delta}_{\mathrm{CE}}[m] = \mathbf{P}[m] - \mathbf{Y}[m]$.

For minibatch m, let

$$\begin{aligned}
\mathbf{H}^{(k-1)}[m] &= \big[\mathbf{h}^{(k-1)}[mB+1], \ldots, \mathbf{h}^{(k-1)}[(m+1)B]\big], \\
\mathbf{G}^{(k)}[m] &= \big[g^{(k)}(\mathbf{h}^{(k)}[mB+1]), \ldots, g^{(k)}(\mathbf{h}^{(k)}[(m+1)B])\big], \\
\mathbf{\Delta}_{\mathrm{CE}}[m] &= \big[\boldsymbol{\delta}_{\mathrm{CE}}[mB+1], \ldots, \boldsymbol{\delta}_{\mathrm{CE}}[(m+1)B]\big],
\end{aligned}$$

where $\boldsymbol{\delta}_{\mathrm{CE}}[n] = \mathbf{p}[n] - \mathbf{y}[n]$. Treating $\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m-1]$ as a fixed state when differentiating the current minibatch objective, define

$$\alpha := \frac{1-\lambda}{B}, \qquad \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m] = \lambda \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m-1] + \alpha\, \mathbf{G}^{(k)}[m]\mathbf{\Delta}_{\mathrm{CE}}[m]^T,$$

and define

$$\mathcal{J}_{\mathrm{CE}}^{(k)}[m] = \frac{1}{2}\left\|\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]\right\|_F^2 = \frac{1}{2}\,\mathrm{Tr}\Big(\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]^T \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]\Big).$$

Hence, for a hidden-layer weight $W_{ij}^{(k)}$,

$$\begin{aligned}
\frac{\partial \mathcal{J}_{\mathrm{CE}}^{(k)}[m]}{\partial W_{ij}^{(k)}} &= \mathrm{Tr}\left(\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]^T \frac{\partial \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]}{\partial W_{ij}^{(k)}}\right) \\
&= \alpha\,\mathrm{Tr}\left(\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]^T \frac{\partial \mathbf{G}^{(k)}[m]}{\partial W_{ij}^{(k)}} \mathbf{\Delta}_{\mathrm{CE}}[m]^T\right) + \alpha\,\mathrm{Tr}\left(\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]^T \mathbf{G}^{(k)}[m] \frac{\partial \mathbf{\Delta}_{\mathrm{CE}}[m]^T}{\partial W_{ij}^{(k)}}\right).
\end{aligned}$$

The first gradient component contains derivatives of the transformed layer-k activations with respect to $W_{ij}^{(k)}$, whereas the second contains $\partial \mathbf{\Delta}_{\mathrm{CE}}[m]/\partial W_{ij}^{(k)}$ and therefore depends on how the logits change through all deeper layers. Indeed, because the label vector is constant with respect to the hidden-layer parameters,

$$\frac{\partial \boldsymbol{\delta}_{\mathrm{CE}}[n]}{\partial W_{ij}^{(k)}} = \frac{\partial \mathbf{p}[n]}{\partial W_{ij}^{(k)}} = \big(\mathrm{diag}(\mathbf{p}[n]) - \mathbf{p}[n]\mathbf{p}[n]^T\big)\,\frac{\partial \mathbf{a}[n]}{\partial W_{ij}^{(k)}},$$

and $\partial \mathbf{a}[n]/\partial W_{ij}^{(k)}$ expands into a chain of derivatives through layers $k+1, \ldots, L$. Retaining this component would therefore reintroduce layer-by-layer propagation from the output. The practical SBD rule retains only the activation-derivative component, the score-based counterpart of the localized EBD update (Erdogan et al., 2025).

Define the layerwise projected-score matrix

$$\mathbf{Q}_{\mathrm{CE}}^{(k)}[m] := \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}[m]\mathbf{\Delta}_{\mathrm{CE}}[m] = \big[\mathbf{q}_{\mathrm{CE}}^{(k)}[mB+1], \ldots, \mathbf{q}_{\mathrm{CE}}^{(k)}[(m+1)B]\big].$$

Using cyclicity of the trace, the retained activation-derivative component becomes

$$\left.\frac{\partial \mathcal{J}_{\mathrm{CE}}^{(k)}[m]}{\partial W_{ij}^{(k)}}\right|_{\mathrm{local}} = \alpha \operatorname{Tr}\left(\mathbf{Q}_{\mathrm{CE}}^{(k)}[m]\frac{\partial \mathbf{G}^{(k)}[m]^T}{\partial W_{ij}^{(k)}}\right).$$

Let $\mathbf{e}_i \in \mathbb{R}^{N^{(k)}}$ denote the ith canonical basis vector. Then

$$\frac{\partial \mathbf{G}^{(k)}[m]}{\partial W_{ij}^{(k)}} = \mathbf{e}_i \begin{bmatrix} g_i'^{(k)}\left(h_i^{(k)}[mB+1]\right) f'^{(k)}\left(u_i^{(k)}[mB+1]\right) h_j^{(k-1)}[mB+1] \\ \vdots \\ g_i'^{(k)}\left(h_i^{(k)}[(m+1)B]\right) f'^{(k)}\left(u_i^{(k)}[(m+1)B]\right) h_j^{(k-1)}[(m+1)B] \end{bmatrix}^T.$$

Substituting this derivative yields

$$\left.\frac{\partial \mathcal{J}_{\mathrm{CE}}^{(k)}[m]}{\partial W_{ij}^{(k)}}\right|_{\mathrm{local}} = \alpha \sum_{n=mB+1}^{(m+1)B} g_i'^{(k)}\left(h_i^{(k)}[n]\right) f'^{(k)}\left(u_i^{(k)}[n]\right) q_{\mathrm{CE},i}^{(k)}[n] h_j^{(k-1)}[n].$$

An identical calculation for the bias gives

$$\left.\frac{\partial \mathcal{J}_{\mathrm{CE}}^{(k)}[m]}{\partial b_i^{(k)}}\right|_{\mathrm{local}} = \alpha \sum_{n=mB+1}^{(m+1)B} g_i'^{(k)}\left(h_i^{(k)}[n]\right) f'^{(k)}\left(u_i^{(k)}[n]\right) q_{\mathrm{CE},i}^{(k)}[n].$$

To write the update compactly, define the derivative matrices

$$\mathbf{G}_d^{(k)}[m] = \left[\mathbf{g}'^{(k)}(\mathbf{h}^{(k)}[mB+1]), \ldots, \mathbf{g}'^{(k)}(\mathbf{h}^{(k)}[(m+1)B])\right],$$
$$\mathbf{F}_d^{(k)}[m] = \left[\mathbf{f}'^{(k)}(\mathbf{u}^{(k)}[mB+1]), \ldots, \mathbf{f}'^{(k)}(\mathbf{u}^{(k)}[(m+1)B])\right],$$

where the derivatives are applied elementwise, and set

$$\mathbf{Z}_{\mathrm{CE}}^{(k)}[m] = \mathbf{G}_d^{(k)}[m] \odot \mathbf{F}_d^{(k)}[m] \odot \mathbf{Q}_{\mathrm{CE}}^{(k)}[m].$$

Then the retained minibatch-local gradients are

$$\nabla_{\mathbf{W}^{(k)}} \mathcal{J}_{\mathrm{CE}}^{(k)}[m]\Big|_{\mathrm{local}} = \alpha\, \mathbf{Z}_{\mathrm{CE}}^{(k)}[m]\mathbf{H}^{(k-1)}[m]^T, \qquad \nabla_{\mathbf{b}^{(k)}} \mathcal{J}_{\mathrm{CE}}^{(k)}[m]\Big|_{\mathrm{local}} = \alpha\, \mathbf{Z}_{\mathrm{CE}}^{(k)}[m]\mathbf{1}_{B\times 1}.$$

For gradient descent with learning rate $\eta > 0$, let $\Delta\mathbf{W}^{(k)}$ denote the parameter increment and set $\zeta = -\eta(1-\lambda)$. Adaptive optimizers act on local directions, with implementation-specific rescaling described in the experimental appendices. The single-sample gradient-descent rule ($B = 1$) is

$$\Delta W_{ij}^{(k)}[n] = \zeta\, g_i'^{(k)}\left(h_i^{(k)}[n]\right) f'^{(k)}\left(u_i^{(k)}[n]\right) q_{\mathrm{CE},i}^{(k)}[n] h_j^{(k-1)}[n],$$

with the bias update obtained by omitting the presynaptic factor $h_j^{(k-1)}[n]$. This is exactly the cross entropy SBD rule reported in Section 3.

For general B, the expression for the update can be written as

$$\Delta W_{ij}^{(k)}[m] = \frac{\zeta}{B} \sum_{n=mB+1}^{mB+B} g_i'^{(k)}\left(h_i^{(k)}[n]\right) f'^{(k)}\left(u_i^{(k)}[n]\right) q_{\mathrm{CE},i}^{(k)}[n] h_j^{(k-1)}[n]. \tag{4}$$

C.3 Extension to convolutional layers

The MLP derivation of the previous subsection extends to convolutional architectures by using every channel–location activation as a function entering the stochastic orthogonality condition and retaining only its dependence on the corresponding filter coefficients. This is the score-based counterpart of the convolutional EBD construction under MSE (Erdogan et al., 2025, Appendix D).

Let $\mathbf{H}^{(k)} \in \mathbb{R}^{P^{(k)}\times M^{(k)}\times N^{(k)}}$ denote the output of the kth convolutional layer, where $P^{(k)}$ is the number of channels and $M^{(k)} \times N^{(k)}$ is the spatial map size. For channel p, the filter tensor is $\mathbf{W}^{(k,p)} \in \mathbb{R}^{P^{(k-1)}\times \Omega^{(k)}\times \Omega^{(k)}}$ with bias $b^{(k,p)} \in \mathbb{R}$, and the convolutional layer output is

$$\mathbf{H}^{(k,p)} = f(\mathcal{U}^{(k,p)}), \qquad \mathcal{U}^{(k,p)} = \left(\mathbf{H}^{(k-1)} * \mathbf{W}^{(k,p)}\right) + b^{(k,p)},$$

where $*$ denotes the channel-summed spatial cross-correlation used by standard CNN implementations, with the kernel left unflipped.

Score-broadcast objective. Let $\boldsymbol{\varepsilon} \in \mathbb{R}^{D_\varepsilon}$ denote the signal broadcast to the hidden convolutional layers. For plain SBD, $\boldsymbol{\varepsilon} = \boldsymbol{\delta}$ and $D_\varepsilon = D_{\text{out}}$; with M-block score expansion, $\boldsymbol{\varepsilon} = \widetilde{\boldsymbol{\delta}}$ and $D_\varepsilon = MD_{\text{out}}$ (Appendix D). For each channel p and each spatial location (r, s), the feature-by-broadcast cross-correlation is

$$\mathbf{R}_{\mathbf{g}\boldsymbol{\varepsilon}}^{(k,p)}[r, s, q] = \mathbb{E}\big[\, g^{(k)}\big(\mathbf{H}^{(k,p)}[r, s]\big)\, \varepsilon_q \,\big], \qquad q = 1, \ldots, D_\varepsilon.$$

The conditional-mean-zero property $\mathbb{E}[\boldsymbol{\varepsilon}^\star \mid X] = \mathbf{0}$ established for both the raw and expanded score signals implies that this cross-correlation vanishes at the population optimum. The per-channel score-broadcast objective at minibatch m is

$$\mathcal{J}_{\text{Score}}^{(k,p)}[m] = \frac{1}{2} \sum_{q=1}^{D_\varepsilon} \big\| \widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\varepsilon}}^{(k,p)}[m, :, :, q] \big\|_F^2,$$

and the layer objective is the sum over $p = 1, \ldots, P^{(k)}$. The exponentially weighted correlation estimate is explicitly

$$\begin{aligned} &\widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\varepsilon}}^{(k,p)}[m, r, s, q] \\ &\quad = \lambda \widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\varepsilon}}^{(k,p)}[m-1, r, s, q] + \frac{1-\lambda}{B} \sum_{n=mB+1}^{(m+1)B} g^{(k)}\Big(\mathbf{H}^{(k,p)}[n, r, s]\Big)\, \varepsilon_q[n]. \end{aligned}$$

Weight update. As in the MLP case, we omit the derivative of the broadcast score through deeper layers and retain only the local dependence of $g^{(k)}(\mathbf{H}^{(k,p)})$ on the filter coefficients. Define the per-sample postsynaptic modulator

$$\boldsymbol{\phi}[n, p, :, :] = \sum_{q=1}^{D_\varepsilon} \varepsilon_q[n] \cdot \Big(\widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\varepsilon}}^{(k,p)}[m, :, :, q] \odot {g^{(k)}}'\big(\mathbf{H}^{(k,p)}[n, :, :]\big) \odot f'\big(\mathcal{U}^{(k,p)}[n, :, :]\big) \Big).$$

Let s_k, d_k, and ρ_k denote the stride, dilation, and padding of layer k, respectively. Under the cross-correlation convention above and using zero-based spatial and kernel indices, the required local derivative is

$$\begin{aligned} &\frac{\partial \mathbf{H}^{(k,p)}[n, r, c]}{\partial \mathbf{W}_{h,i,j}^{(k,p)}} \\ &\quad = f'\Big(\mathcal{U}^{(k,p)}[n, r, c]\Big)\, \mathbf{H}^{(k-1,h)}\big[n,\ s_k r + d_k i - \rho_k,\ s_k c + d_k j - \rho_k\big]. \end{aligned}$$

Substituting this expression into the correlation-objective derivative gives the filter increment

$$\Delta \mathbf{W}_{h,i,j}^{(k,p)} = \frac{\zeta}{B} \sum_{n=mB+1}^{(m+1)B} \sum_{r,c} \boldsymbol{\phi}[n, p, r, c]\, \mathbf{H}^{(k-1,h)}\big[n,\ s_k r + d_k i - \rho_k,\ s_k c + d_k j - \rho_k\big],$$

where the signed coefficient ζ absorbs the descent sign and the factor $1 - \lambda$, as in the previous subsection. Locations outside the input map are interpreted according to the layer's padding rule (and are zero under zero padding). Thus, for unit stride and dilation with same padding, the presynaptic factor paired with output location (r, c) and kernel offset (i, j) is precisely $\mathbf{H}^{(k-1,h)}[n, r+i-\rho_k, c+j-\rho_k]$. The corresponding bias increment is

$$\Delta b^{(k,p)} = \frac{\zeta}{B} \sum_{n=mB+1}^{(m+1)B} \sum_{r,s} \boldsymbol{\phi}[n, p, r, s].$$

Setting $\boldsymbol{\varepsilon}$ to the MSE output error recovers the convolutional EBD update (Erdogan et al., 2025, Appendix D.1).

Three-factor structure. The convolutional update preserves the three-factor structure of the MLP case: for each filter coefficient, the presynaptic factor is $\mathbf{H}^{(k-1,h)}$ evaluated at the stride-, dilation-, and padding-adjusted input coordinate defined above; $\big(g^{(k)}\big)' \odot f'$ is the postsynaptic sensitivity, and the score-projected term $\sum_q \varepsilon_q \cdot \widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\varepsilon}}^{(k,p)}$ is the broadcast modulator. The same three-factor structure depicted in Figure 1a therefore applies to convolutional layers, with the convolution operator replacing the matrix product of the MLP case.

Auxiliary regularization. The convolutional weight-entropy regularizer of Erdogan et al. (2025, Appendix D.1.2) is disabled in the reported CNN runs ($c_{\mathrm{cov}}^{(k)} = 0$). Active fully connected regularizers are specified in Appendices E.1 and F.

C.4 Update complexity

We summarize the per-minibatch update cost of the rule in Table 3 and compare it to backpropagation (BP) and Direct Feedback Alignment (DFA) under a fully-connected layer model. Let L denote the number of layers, $N^{(k)}$ the width of layer k (so $N^{(0)}$ is the input dimension and $N^{(L)} = D$ the number of classes), B the minibatch size, and D the output-score dimension. The hidden-layer broadcast dimension is $D_\varepsilon = D$ for plain SBD and $D_\varepsilon = MD$ for the M-block score-expansion variant of Section 6.

The forward pass is shared by all three methods and costs $\mathcal{O}\big(B \sum_{k=1}^{L} N^{(k)} N^{(k-1)}\big)$. The local outer-product weight update at layer k, of cost $\mathcal{O}\big(B\, N^{(k)} N^{(k-1)}\big)$, is also identical in form across the three methods. We compare the cost of producing the postsynaptic modulator, excluding shared forward/weight-gradient operations and experiment-specific auxiliary regularization.

BP. The error signal $\boldsymbol{\delta}^{(k)}$ at hidden layer k is obtained by propagating $\boldsymbol{\delta}^{(k+1)}$ through $\mathbf{W}^{(k+1)\,T}$, costing $\mathcal{O}\big(B\, N^{(k)} N^{(k+1)}\big)$ per layer. The total hidden-signal propagation cost is therefore $\mathcal{O}\big(B \sum_{k=1}^{L-1} N^{(k)} N^{(k+1)}\big)$, comparable to the forward pass, and this step requires weight transport.

DFA. The output error of dimension D is broadcast to layer k through a *fixed random* feedback matrix $\mathbf{B}^{(k)} \in \mathbb{R}^{N^{(k)} \times D}$, costing $\mathcal{O}\big(B\, N^{(k)} D\big)$ per layer with $\mathcal{O}\big(N^{(k)} D\big)$ storage. No backward chain is computed.

SBD. The output score, also of dimension D, is broadcast through the *adaptive* correlation matrix $\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)} \in \mathbb{R}^{N^{(k)} \times D}$. Two operations contribute at layer k: the EMA correlation update (Step 2), which is an outer product of cost $\mathcal{O}\big(B\, N^{(k)} D\big)$, and the projection $\mathbf{q}_{\mathrm{CE}}^{(k)} = \widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)} \boldsymbol{\delta}_{\mathrm{CE}}$ (Step 3), of cost $\mathcal{O}\big(B\, N^{(k)} D\big)$. Storage of $\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}$ is $\mathcal{O}\big(N^{(k)} D\big)$, matching DFA.

Comparison. Per layer, the modulator cost is $\mathcal{O}\big(B\, N^{(k)} N^{(k+1)}\big)$ for BP versus $\mathcal{O}\big(B\, N^{(k)} D\big)$ for DFA and SBD. When $D \ll N^{(k+1)}$, both broadcast methods have lower modulator cost. SBD and DFA share the same asymptotic complexity, but SBD performs roughly twice the work of DFA per layer (one EMA outer product plus one projection, versus DFA's single projection through a fixed matrix); in exchange, $\widehat{\mathbf{R}}_{\mathrm{CE}}^{(k)}$ is *learned*. Replacing D by $D_\varepsilon = MD$ in the SBD expressions above gives the cost of the score-expansion variant of Section 6, which scales linearly in the expansion factor M while preserving the orthogonality framework.

Convolutional layers. The same analysis extends to the convolutional layers of Appendix C.3. Let $P^{(k)}$ be the number of output channels at layer k, $\Omega^{(k)}$ the kernel size, and $M^{(k)} \times N^{(k)}$ the spatial output size. The shared forward convolution and local filter-gradient contraction each cost $\mathcal{O}\big(B\, P^{(k)} P^{(k-1)} (\Omega^{(k)})^2 M^{(k)} N^{(k)}\big)$ per layer. The SBD modulator at convolutional layer k requires maintaining a per-channel feature-by-broadcast correlation tensor $\widehat{\mathbf{R}}_{\mathbf{g}\boldsymbol{\varepsilon}}^{(k,p)} \in \mathbb{R}^{M^{(k)} \times N^{(k)} \times D_\varepsilon}$, where $D_\varepsilon = D$ for plain SBD and $D_\varepsilon = MD$ for M-block score expansion. Its total size is $\mathcal{O}\big(P^{(k)} D_\varepsilon M^{(k)} N^{(k)}\big)$, with the EMA update and the projection each costing $\mathcal{O}\big(B\, P^{(k)} D_\varepsilon M^{(k)} N^{(k)}\big)$. For a following convolution, BP instead propagates its signal through layer $k+1$, with cost

$$\mathcal{O}\Big(B P^{(k)} P^{(k+1)} (\Omega^{(k+1)})^2 M^{(k+1)} N^{(k+1)}\Big).$$

The broadcast cost scales with D_ε at the current spatial resolution, whereas BP scales with the next layer's fan-out, kernel area, and output resolution. Score expansion increases correlation storage and update/projection work linearly in M; modulator construction is a separate cost.

These asymptotic operation counts are independent of a particular software stack and do not directly predict realized runtime or energy consumption. Empirical convergence and GPU-energy measurements for the Tiny ImageNet implementation are reported with the accuracy curves in Appendix F.4.

D SCORE VECTOR EXPANSION: GENERAL THEORY AND CROSS ENTROPY EXAMPLE

This appendix provides the full details behind the main-text summary in Section 6. We first develop the score vector expansion for a general differentiable loss whose population-optimal score has conditional mean zero, and then specialize the construction to cross entropy.

D.1 GENERAL FORMULATION

Let $\mathcal{L}(\mathbf{y}, \mathbf{a})$ be a differentiable loss with output score $\boldsymbol{\delta}(\mathbf{y}, \mathbf{a}) = \nabla_{\mathbf{a}}\mathcal{L}(\mathbf{y}, \mathbf{a}) \in \mathbb{R}^{D_{\text{out}}}$, and assume that the population-optimal predictor $\mathbf{a}^\star(X)$ satisfies the conditional mean-zero property

$$\mathbb{E}\big[\boldsymbol{\delta}(\mathbf{Y}, \mathbf{a}^\star(X)) \,\big|\, X\big] = \mathbf{0},$$

so that the orthogonality machinery of Theorem 3 applies. The layerwise correlation matrix

$$\mathbf{R}^{(k)} = \mathbb{E}\big[g^{(k)}(\mathbf{h}^{(k)})\boldsymbol{\delta}^T\big] \in \mathbb{R}^{d_k \times D_{\text{out}}}$$

has column rank at most D_{out}, regardless of the hidden-layer width d_k. When d_k greatly exceeds D_{out}, this caps the number of independent decorrelation constraints and motivates enlarging the broadcast signal.

Modulated scores preserve conditional mean-zero. Let $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^{D_{\text{out}}}$ be any measurable vector-valued modulator of the input, and define the *modulated score* at the population optimum by

$$\boldsymbol{\eta}^{\boldsymbol{\phi},\star}(X) \;:=\; \boldsymbol{\phi}(X) \odot \boldsymbol{\delta}\big(\mathbf{Y}, \mathbf{a}^\star(X)\big),$$

where $\odot$ denotes the Hadamard product. Assume the modulated score is integrable. Because $\boldsymbol{\phi}(X)$ is measurable with respect to X, it can be pulled out of the conditional expectation. Hence

$$\mathbb{E}\big[\boldsymbol{\eta}^{\boldsymbol{\phi},\star}(X) \,\big|\, X\big] = \boldsymbol{\phi}(X) \odot \mathbb{E}\big[\boldsymbol{\delta}(\mathbf{Y}, \mathbf{a}^\star(X)) \,\big|\, X\big] = \mathbf{0}.$$

Applying the tower-property orthogonality lemma (Lemma 1), we conclude

$$\mathbb{E}\big[\boldsymbol{\eta}^{\boldsymbol{\phi},\star}(X)g(X)^T\big] = \mathbf{0} \quad \text{whenever } \mathbb{E}\|\boldsymbol{\eta}^{\boldsymbol{\phi},\star}(X)g(X)^T\| < \infty.$$

Under these integrability conditions, any X-measurable Hadamard modulator preserves population-optimal orthogonality. Modulators can use the raw input, intermediate activations, or output predictions. Forward randomness independent of Y conditional on X can be included by conditioning jointly on X and that randomness.

Broadcast expansion by stacking modulators. Given M X-measurable modulators $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_M$, let $D_\varepsilon := MD_{\text{out}}$ and define the *expanded score* during training by

$$\tilde{\boldsymbol{\delta}}(X) \;:=\; \begin{bmatrix} \boldsymbol{\phi}_1(X) \odot \boldsymbol{\delta}(\mathbf{Y}, \mathbf{a}(X)) \\ \boldsymbol{\phi}_2(X) \odot \boldsymbol{\delta}(\mathbf{Y}, \mathbf{a}(X)) \\ \vdots \\ \boldsymbol{\phi}_M(X) \odot \boldsymbol{\delta}(\mathbf{Y}, \mathbf{a}(X)) \end{bmatrix} \in \mathbb{R}^{D_\varepsilon},$$

obtained by stacking the M modulated scores. Replacing $\mathbf{a}(X)$ by the population optimum $\mathbf{a}^\star(X)$, the argument above shows that each block of length D_{out} satisfies the conditional mean-zero property, and hence

$$\mathbb{E}\Big[\tilde{\boldsymbol{\delta}}^\star(X) \,\Big|\, X\Big] = \mathbf{0}, \qquad \mathbb{E}\Big[\tilde{\boldsymbol{\delta}}^\star(X)g^{(k)}(\mathbf{h}^{(k)}(X))^T\Big] = \mathbf{0}. \tag{5}$$

The expanded correlation matrix

$$\tilde{\mathbf{R}}^{(k)} \;=\; \mathbb{E}\Big[g^{(k)}(\mathbf{h}^{(k)})\,\tilde{\boldsymbol{\delta}}^T\Big] \in \mathbb{R}^{d_k \times D_\varepsilon}$$

now has column rank at most $D_\varepsilon = MD_{\text{out}}$, providing the layerwise decorrelation objective $\tilde{\mathcal{J}}^{(k)} = \frac{1}{2}\|\tilde{\mathbf{R}}^{(k)}\|_F^2$ with up to M times more independent directions along which to suppress dependence between the hidden representation and the broadcast score, at the cost of a linear increase in stored correlation state.

Algorithmic formulation. The expanded-score procedure is identical to plain SBD except that the D_{out}-dimensional score and its batch matrix are replaced by their D_ε-dimensional counterparts:

$$\begin{aligned}
\tilde{\boldsymbol{\delta}}[n] &= \big[\boldsymbol{\phi}_1(\mathbf{x}[n]) \odot \boldsymbol{\delta}(\mathbf{y}[n], \mathbf{a}(\mathbf{x}[n])); \ \ldots; \ \boldsymbol{\phi}_M(\mathbf{x}[n]) \odot \boldsymbol{\delta}(\mathbf{y}[n], \mathbf{a}(\mathbf{x}[n]))\big], \\
\tilde{\boldsymbol{\Delta}}[m] &= \big[\tilde{\boldsymbol{\delta}}[mB+1] \quad \cdots \quad \tilde{\boldsymbol{\delta}}[mB+B]\big] \in \mathbb{R}^{D_\varepsilon \times B}, \\
\widehat{\widetilde{\mathbf{R}}}^{(k)}[m] &= \lambda \widehat{\widetilde{\mathbf{R}}}^{(k)}[m-1] + \frac{1-\lambda}{B} \mathbf{G}^{(k)}[m] \, \tilde{\boldsymbol{\Delta}}[m]^T, \\
\tilde{\mathbf{q}}^{(k)}[mB+n] &= \widehat{\widetilde{\mathbf{R}}}^{(k)}[m] \, \tilde{\boldsymbol{\delta}}[mB+n], \text{ for } n = 1, \ldots, B,
\end{aligned}$$

$$\Delta W_{ij}^{(k)}[m] = \frac{\zeta}{B} \sum_{l=mB+1}^{mB+B} g_i'^{(k)}\big(h_i^{(k)}[l]\big) \, f'^{(k)}\big(u_i^{(k)}[l]\big) \, \tilde{q}_i^{(k)}[l] \, h_j^{(k-1)}[l]. \tag{6}$$

The output layer continues to use the raw D_{out}-dimensional score for its standard gradient update.

D.2 Cross entropy instantiation

For cross entropy classification, the output dimension is $D_{\text{out}} = D$ (the number of classes) and the output score is $\boldsymbol{\delta}_{\text{CE}} = \mathbf{p} - \mathbf{y}$, with $\mathbf{p}^\star(X) = \mathbf{q}(X)$ as the population minimizer. The conditional mean-zero property required by the general construction is established in Theorem 1, so the modulated and expanded score machinery above applies to any deterministic X-measurable modulator. In our experiments, the modulators are built from the predictive distribution $\mathbf{p}(X)$, which is itself a deterministic function of X for fixed network parameters.

For classification networks in which the hidden width d_k greatly exceeds D; for example, CIFAR-10 with $D = 10$ and $d_k \in \{10^3, 10^5\}$ in the fully-connected and convolutional layers, respectively, the rank bottleneck in $\mathbf{R}_{\text{CE}}^{(k)} \in \mathbb{R}^{d_k \times D}$ is severe, and score expansion provides a natural remedy.

A concrete $3D$-dimensional instance. For the CNN experiments in Section 7 we adopt a minimal $M = 3$ instance built from two natural X-measurable modulators obtained from $\mathbf{p}(X)$, together with the identity. Let $\mathbf{1}_D \in \mathbb{R}^D$ denote the all-ones vector, and let $\text{roll}_k(\mathbf{v})$ denote the cyclic shift of $\mathbf{v}$ by k positions, that is $[\text{roll}_k(\mathbf{v})]_d = v_{((d-k-1) \bmod D)+1}$. The three modulators are

$$\boldsymbol{\phi}_1(X) = \mathbf{1}_D, \qquad \boldsymbol{\phi}_2(X) = \mathbf{p}(X), \qquad \boldsymbol{\phi}_3(X) = \text{roll}_k(\mathbf{p}(X)),$$

where $k = 5$ in the reported CNN configuration. Class indices are arbitrary, so the shift does not represent semantic proximity. The resulting expanded score is

$$\tilde{\boldsymbol{\delta}}_{\text{CE}} \ = \ \begin{bmatrix} \boldsymbol{\delta}_{\text{CE}} \\ \mathbf{p} \odot \boldsymbol{\delta}_{\text{CE}} \\ \text{roll}_k(\mathbf{p}) \odot \boldsymbol{\delta}_{\text{CE}} \end{bmatrix} \in \mathbb{R}^{3D}.$$

Block one is the raw cross-entropy score. Block two weights each coordinate by its predicted class probability, and block three uses a cyclically shifted probability vector. These input-dependent weightings can supply distinct score–feature correlations. All three modulators are deterministic functions of X through the forward pass, so (5) applies and the expanded decorrelation condition remains an exact orthogonality relation at the population optimum.

Algorithm. Specializing the general expanded-SBD procedure above to the cross entropy case yields the algorithm in Table 4. The only modifications from Appendix C, Table 3 are Step 2, in which $\boldsymbol{\Delta}_{\text{CE}}[m]$ is replaced by $\tilde{\boldsymbol{\Delta}}_{\text{CE}}[m] \in \mathbb{R}^{D_\varepsilon \times B}$, where $D_\varepsilon = 3D$, and Step 3, in which the projection uses $\widehat{\widetilde{\mathbf{R}}}_{\text{CE}}^{(k)}$ and $\tilde{\boldsymbol{\delta}}_{\text{CE}}$.

E CIFAR-10 experimental setup

This appendix documents the codebase used for the CIFAR-10 broadcast-learning experiments reported in the main text. The expanded SBD configuration is implemented by the per-seed launch scripts `run_cemseN_s*.sh`, which call `CIFAR10_CEMSEN.py` with cross-entropy loss,

Table 4: Cross entropy SBD with $3D$-dimensional score expansion. Differences from Appendix C, Table 3 are shown in italics.

Step	Operation
1	Forward pass to obtain $\mathbf{h}^{(k)}$, $\mathbf{a}$, $\mathbf{p} = \mathrm{softmax}(\mathbf{a})$, and the score $\boldsymbol{\delta}_{\mathrm{CE}} = \mathbf{p} - \mathbf{y}$.
2	*Form the expanded score* $\tilde{\boldsymbol{\delta}}_{\mathrm{CE}} = [\boldsymbol{\delta}_{\mathrm{CE}};\ \mathbf{p} \odot \boldsymbol{\delta}_{\mathrm{CE}};\ \mathrm{roll}_k(\mathbf{p}) \odot \boldsymbol{\delta}_{\mathrm{CE}}] \in \mathbb{R}^{D_\varepsilon}$, *where* $D_\varepsilon = 3D$*, and stack it into the batch matrix* $\tilde{\boldsymbol{\Delta}}_{\mathrm{CE}}[m] \in \mathbb{R}^{D_\varepsilon \times B}$. *Update the expanded correlation estimate* $\widehat{\widetilde{\mathbf{R}}}^{(k)}_{\mathrm{CE}}[m]$.
3	*Project the broadcast score to hidden layer* k *by* $\tilde{\mathbf{q}}^{(k)}_{\mathrm{CE}}[n] = \widehat{\widetilde{\mathbf{R}}}^{(k)}_{\mathrm{CE}}[m]\, \tilde{\boldsymbol{\delta}}_{\mathrm{CE}}[n]$ *for* $n = mB + 1, \ldots, (m+1)B$.
4	For $B = 1$, the weight increment is $\Delta W^{(k)}_{ij}[n] = \zeta\, g_i'^{(k)}(h^{(k)}_i[n])\, f'^{(k)}(u^{(k)}_i[n])\, \tilde{q}^{(k)}_{\mathrm{CE},i}[n]\, h^{(k-1)}_j[n]$, with analogous bias updates. For $B > 1$, use Eq. (6).
5	Update the output layer with the ordinary D-dimensional cross-entropy gradient.

`method=sbd`, and `err_expand=2`. This setting uses the $3D$-dimensional score expansion, so that the broadcast dimension is $3D = 30$ for CIFAR-10. The main-text result is averaged over five independent seed runs using the 5-layer CNN described in Section E.2. All CIFAR-10 experiments run in `float32` on a single NVIDIA V100 (32GB) GPU using PyTorch. For the reported baseline-width CIFAR-10 experiments, the approximate times to complete 200 epochs were 75, 91, 96, and 101 minutes for BP, DFA, SBD, and SBD with score expansion, respectively. For the reported $4\times$-width CIFAR-10 experiments, the approximate times to complete 200 epochs were 170 and 420 minutes for BP and SBD with score expansion, respectively.

For CIFAR-10, we use the standard dataset partition of 50,000 training images and 10,000 test images, and all reported CIFAR-10 accuracies are top-1 accuracies on the official test split. The architecture follows the original EBD CNN setup, with cross entropy replacing MSE; method-specific optimization settings are described below. For the expanded-SBD variant, Appendix E.5 documents the chosen score-expansion modulator set together with the multi-seed ablation comparing it with alternative expansions.

PyTorch (Paszke et al., 2019) is distributed under the BSD-3-Clause license, while the CIFAR-10 dataset (Krizhevsky, 2009) is publicly released by the University of Toronto for research use without a formal license declaration on its official distribution page; both are used here in accordance with their stated terms.

E.1 Score broadcast and decorrelation

SBD replaces the global backpropagation chain with local, per-layer learning rules driven by a broadcast score signal. At each mini-batch, the forward pass produces pre-softmax logits $\mathbf{a} \in \mathbb{R}^D$ and probabilities $\mathbf{p} = \mathrm{softmax}(\mathbf{a}/T)$, where $D = 10$ for CIFAR-10 and T is the implementation variable `SOFTMAX_TEMP`. In cross-entropy mode, the implementation uses the broadcast score

$$\boldsymbol{\delta} = \alpha(\mathbf{p} - \mathbf{y}) \in \mathbb{R}^D, \qquad \alpha = \texttt{ERR_SCALE},$$

where α is the score scaling and $\mathbf{y}$ is the one-hot label vector. The reported runs use $T = \alpha = 1$, so this is the cross-entropy derivative with respect to the logits.

For hidden-layer SBD updates, this D-dimensional signal is expanded into a higher-dimensional broadcast vector $\boldsymbol{\varepsilon} \in \mathbb{R}^{D_\varepsilon}$ by concatenating blocks of the form $\boldsymbol{\phi}_\ell(\mathbf{x}) \odot \boldsymbol{\delta}$, where each $\boldsymbol{\phi}_\ell$ is an X-measurable deterministic function of the input $\mathbf{x}$ (in the reported setting, through $\mathbf{p}$ and $\mathbf{a}$). In the main expanded-SBD configuration, implemented with `err_expand=2`, the hidden-layer broadcast vector is

$$\boldsymbol{\varepsilon}(\mathbf{x}, \mathbf{y}) = \big[\boldsymbol{\delta};\ \mathbf{p} \odot \boldsymbol{\delta};\ \mathrm{roll}_k(\mathbf{p}) \odot \boldsymbol{\delta}\big] \in \mathbb{R}^{D_\varepsilon}, \qquad D_\varepsilon = 3D = 30. \tag{7}$$

Thus the output layer receives the raw 10-dimensional score $\boldsymbol{\delta}$, while the hidden layers receive the expanded $3D$-dimensional broadcast signal, with $k = 5$ in the reported configuration. Since each

modulator is deterministic conditional on X, the conditional mean-zero property is preserved at the Bayes optimum:

$$\mathbb{E}[\boldsymbol{\phi}_\ell(X) \odot \boldsymbol{\delta}^\star \mid X] = \boldsymbol{\phi}_\ell(X) \odot \mathbb{E}[\boldsymbol{\delta}^\star \mid X] = 0.$$

Thus score expansion preserves the population orthogonality property used by SBD.

For each hidden trainable layer k with pre-activation $\mathbf{u}^{(k)} = W^{(k)}\mathbf{h}^{(k-1)}$, post-activation $\mathbf{h}^{(k)} = \mathrm{ReLU}(\mathbf{u}^{(k)})$, and incoming activation $\mathbf{h}^{(k-1)}$, we use the feature-by-broadcast orientation adopted in the main text, $\widehat{\mathbf{R}}^{(k)}_{g\varepsilon} \in \mathbb{R}^{N^{(k)} \times D_\varepsilon}$. If an implementation stores the transpose for memory-layout convenience, it represents the same object as $\widehat{\mathbf{R}}^{(k)}_{\varepsilon g} = \widehat{\mathbf{R}}^{(k)T}_{g\varepsilon}$; all equations below use the feature-by-broadcast orientation. The CNN uses the identity feature map $g(\mathbf{h}) = \mathbf{h}$. SBD maintains two mini-batch EMAs:

$$\widehat{\mathbf{R}}^{(k)}_{g\varepsilon}[m] \leftarrow \lambda\, \widehat{\mathbf{R}}^{(k)}_{g\varepsilon}[m-1] + (1-\lambda) \cdot \tfrac{1}{B} \sum_{l=mB+1}^{mB+B} \mathbf{h}^{(k)}[l]\, \boldsymbol{\varepsilon}[l]^T \qquad \text{(cross-correlation)}$$

$$\widehat{\mathbf{R}}^{(k)}_{hh}[m] \leftarrow \lambda_2\, \widehat{\mathbf{R}}^{(k)}_{hh}[m-1] + (1-\lambda_2) \cdot \tfrac{1}{B} \sum_{l=mB+1}^{mB+B} \mathbf{h}^{(k)}[l]\mathbf{h}^{(k)}[l]^T \quad \text{(auto-correlation; FC only)}$$

with decay rates $\lambda =$ `Reh_lambda` and $\lambda_2 =$ `Reh_lambda2`. Both decays are annealed toward 1 at the start of each epoch from zero-based epoch 2 onward according to $\lambda \leftarrow \lambda + \rho(1-\lambda)$ with $\rho =$ `Reh_lambda_drop`.

The tensor supplied to Adam at layer k is a weighted sum of three local terms:

$$G^{(k)} \;=\; c^{(k)}_{\text{score}}\, G^{\text{score}}_{W^{(k)}} \;+\; c^{(k)}_{\text{cov}}\, G^{\text{cov}}_{W^{(k)}} \;+\; c^{(k)}_{\ell_1}\, G^{\ell_1}_{W^{(k)}}.$$

The coefficient $c^{(k)}_{\text{score}}$ denotes the weight of the score-broadcast decorrelation term. In the implementation, this coefficient corresponds to the `CMSE_*` hyperparameters, retaining the naming convention of the original MSE/EBD codebase. Adam applies its moment estimates, coordinatewise scaling, and weight decay to this tensor before subtracting the resulting parameter step. The three terms are:

- **Three-factor score-broadcast update**

  $$G^{\text{score}}_{W^{(k)}} \;=\; \tfrac{1}{B} \sum_{l \in \mathcal{B}} \big[\, f'(\mathbf{u}^{(k)}[l]) \odot \big(\widehat{\mathbf{R}}^{(k)}_{g\varepsilon}\, \boldsymbol{\varepsilon}[l]\big) \,\big]\, \mathbf{h}^{(k-1)}[l]^T.$$

  This is the product of three factors: the ReLU derivative $f'(\mathbf{u}^{(k)})$, the broadcast modulator $\widehat{\mathbf{R}}^{(k)}_{g\varepsilon}\boldsymbol{\varepsilon}$, and the presynaptic activation $\mathbf{h}^{(k-1)}$, accumulated across the batch $\mathcal{B}$. The local direction uses the updated EMA, with scale set by $c^{(k)}_{\text{score}}$. Convolutions use pre-pooling activations and sum the corresponding receptive-field products over spatial locations. At the output layer, $\boldsymbol{\varepsilon}$ is replaced by the raw $\boldsymbol{\delta}$, f' is set to one, and $\widehat{\mathbf{R}}^{(k)}_{g\varepsilon}$ is the identity so that the update reduces to the standard single-layer cross entropy gradient.

- **Activation covariance direction** (FC layers only) (Erdogan et al., 2025; Ozsoy et al., 2022; Bozkurt et al., 2023a;b)

  $$G^{\text{cov}}_{W^{(k)}} \;=\; -\tfrac{2}{B\,D_k} \sum_{l \in \mathcal{B}} \big[\, f'(\mathbf{u}^{(k)}[l]) \odot (\widehat{\mathbf{R}}^{(k)}_{hh} + \eta I)^{-1}(\mathbf{h}^{(k)}[l] + 10^{-5}\mathbf{1}) \,\big]\, \mathbf{h}^{(k-1)}[l]^T,$$

  motivated by the log-determinant activation objective. Here $\eta = 10^{-8}$ (the `Reh_ini` setting) regularizes the inverse and D_k is the layer width. At the classifier, this local rule uses $\mathbf{h}^{(k)} = \mathbf{p}$ and $f' = 1$; convolutional layers use $c^{(k)}_{\text{cov}} = 0$ (`CCOV_HIDDEN`).

- **Activation ℓ_1 sparsity**

  $$G^{\ell_1}_{W^{(k)}} \;=\; \tfrac{1}{B\,D_k} \sum_{l \in \mathcal{B}} \mathrm{sign}(\mathbf{h}^{(k)}[l])\, \mathbf{h}^{(k-1)}[l]^T,$$

  applied only to the hidden fully connected layer fc3 in the reported CIFAR-10 expanded-SBD implementation. It is disabled for the convolutional layers and is not applied to the final classifier fc4.

The scalar coefficients $(c^{(k)}_{\text{score}}, c^{(k)}_{\text{cov}}, c^{(k)}_{\ell_1})$ are given in Table 7. The local SBD terms are computed inside a `torch.no_grad()` context, manually written into `param.grad`, and applied via a standard Adam optimizer. In the SBD runs, hidden-layer updates are supplied by these local broadcast-decorrelation rules rather than by a global backpropagation chain.

E.2 NETWORK ARCHITECTURE

The network is a minimalist 5-layer CNN: three convolutional blocks followed by two fully connected layers. It uses no bias terms, no BatchNorm, no dropout, and no residual connections. Average pooling is used throughout. This minimal choice isolates the SBD update rule from confounds introduced by normalization layers.

Table 5: Baseline CNN architecture. Input: CIFAR-10 images of shape $(3, 32, 32)$. Total trainable parameters: 1,289,600.

Layer	Operation	Output shape	Parameters
conv1	5×5 conv, stride 1, pad 2	(128, 32, 32)	9,600
act1	ReLU	(128, 32, 32)	–
pool1	AvgPool 2×2	(128, 16, 16)	–
conv2	5×5 conv, stride 1, pad 2	(64, 16, 16)	204,800
act2	ReLU	(64, 16, 16)	–
pool2	AvgPool 2×2	(64, 8, 8)	–
conv3	2×2 conv, stride 2, pad 0	(64, 4, 4)	16,384
act3	ReLU	(64, 4, 4)	–
flatten	reshape	(1024,)	–
fc3	linear	(1024,)	1,048,576
act4	ReLU	(1024,)	–
fc4	linear	(10,)	10,240
softmax	softmax at $T=1$	(10,)	–

Layer widths are parameterized via the CLI flags `-P0`, `-P1`, `-P2`, and `-fc_hidden` (default $128, 64, 64, 1024$), which in turn set the channel counts of conv1, conv2, conv3, and fc3 respectively. The fc3 input dimension $P_2 \cdot 4 \cdot 4$ is derived automatically. Weights are initialized from a Gaussian distribution with standard deviation $\sqrt{2}\sqrt{1/\texttt{scale_factor}}/\sqrt{\text{fan_in}}$, giving a scaled variant of He et al.'s fan-in initialization (He et al., 2015) with $\texttt{scale_factor}= 6$. In the reported CIFAR-10 runs, data augmentation consists of per-channel normalization, random cropping after 2-pixel reflection padding, and horizontal flipping. The loader caches an initial random flip for each image and reverses all flip choices on alternating epochs.

For the additional width-scaling experiments discussed in the main text, we also evaluated a $4\times$-width variant of the same CNN.

Table 6: CNN architecture at $4\times$ width. Input: CIFAR-10 images of shape $(3, 32, 32)$. Total trainable parameters: 20,395,520 ($\approx 16\times$ the baseline model count).

Layer	Operation	Output shape	Parameters
conv1	5×5 conv, stride 1, pad 2	(512, 32, 32)	38,400
act1	ReLU	(512, 32, 32)	–
pool1	AvgPool 2×2	(512, 16, 16)	–
conv2	5×5 conv, stride 1, pad 2	(256, 16, 16)	3,276,800
act2	ReLU	(256, 16, 16)	–
pool2	AvgPool 2×2	(256, 8, 8)	–
conv3	2×2 conv, stride 2, pad 0	(256, 4, 4)	262,144
act3	ReLU	(256, 4, 4)	–
flatten	reshape	(4096,)	–
fc3	linear	(4096,)	16,777,216
act4	ReLU	(4096,)	–
fc4	linear	(10,)	40,960
softmax	softmax at $T=1$	(10,)	–

The wider model multiplies all three convolutional channel counts and the hidden FC width by four, preserving spatial sizes and the ten-class output. Most weight matrices therefore grow by a factor of sixteen, accounting for the parameter counts in Tables 5 and 6.

E.3 TRAINING HYPERPARAMETERS

Table 7 lists the hyperparameters used by the expanded-SBD CIFAR-10 launch scripts `run_cemseN_s*.sh`. Unless otherwise noted, expanded-SBD runs inherit these values; control runs use the corresponding values specified in their own launch scripts. The training loop performs 201 passes, indexed 0–200, and skips the last full training batch as well as any incomplete batch. Learning-rate decay begins at the start of zero-based epoch 2. Evaluation uses unaugmented training and test images after every pass.

At baseline width, plain SBD and DFA use $B = 16$ and initial learning rate 10^{-4}, while expanded SBD uses $B = 64$ and 4×10^{-4}. BP uses $(B, \eta_0) = (16, 10^{-4})$ with decay 0.96 at baseline width and $(64, 4 \times 10^{-4})$ with decay 0.97 at $4\times$ width. The wider expanded-SBD run retains the baseline expanded-SBD recipe.

Table 7: CEMSEN training hyperparameters corresponding to the expanded-SBD per-seed launch scripts `run_cemseN_s*.sh`. Column *flag* is the CLI name in the launch script; column *Symbol* names the variable used in the equations of Section E.1.

Group	Flag	Symbol	Value
General	`-logger_name`	seed	5 independent seeds
	`-n_epochs`	–	201
	`-batch_size`	B	64
	`-loss_type`	–	cross entropy
	`-augmentation`	–	1 (crop + flip)
Method	`-method`	–	`sbd`
	`-err_expand`	–	2 (30 coordinates)
	`-ERR_SCALE`	α	1.0
	`-SOFTMAX_TEMP`	T	1.0
	effective D_ε	D_ε	30
Optimizer	`-lr`	η_0	4×10^{-4}
	`-lr_drop_rate`	–	0.97
	`-lr_drop_every`	–	1 epoch
	`-weight_decay`	–	1×10^{-5}
	Adam β_1, β_2	–	0.9, 0.999
	precision	–	`float32`
SBD coefficients	`-CMSE_OUT`	$c_{\text{score}}^{(\text{fc4})}$	10
	`-CMSE_OUT2`	$c_{\text{score}}^{(\text{fc3})}$	0.1
	`-CMSE_HIDDEN`	$c_{\text{score}}^{(\text{conv})}$	0.1
	`-CCOV_OUT`	$c_{\text{cov}}^{(\text{fc4})}$	1×10^{-7}
	`-CCOV_OUT2`	$c_{\text{cov}}^{(\text{fc3})}$	1×10^{-7}
	`-CCOV_HIDDEN`	$c_{\text{cov}}^{(\text{conv})}$	0
	`-CL1_OUT`	$c_{\ell_1}^{(\text{fc3})}$	1×10^{-11}
	`-CL1_HIDDEN`	$c_{\ell_1}^{(\text{hidden})}$	0
Covariance EMA	`-Reh_lambda`	λ	0.99999
	`-Reh_lambda2`	λ_2	0.99999
	`-Reh_lambda_drop`	ρ	0.04
	`-Reh_lambda_drop_every`	–	1 epoch
	`-Reh_gain`	$\text{std}(\widehat{\mathbf{R}}_{g\varepsilon}^{(\text{conv})})$	0.01
	`-Reh_gain_lin`	$\text{std}(\widehat{\mathbf{R}}_{g\varepsilon}^{(\text{FC})})$	0.01
	`-Reh_ini`	init diagonal of $\widehat{\mathbf{R}}_{hh}$	1×10^{-8}
Architecture	`-P0`, `-P1`, `-P2`	channel counts	128, 64, 64
	`-fc_hidden`	fc3 width F_h	1024
	`-use_bias`	–	0 (disabled)
	`-scale_factor`, `-init_dist`	init scale, dist.	6, Gaussian

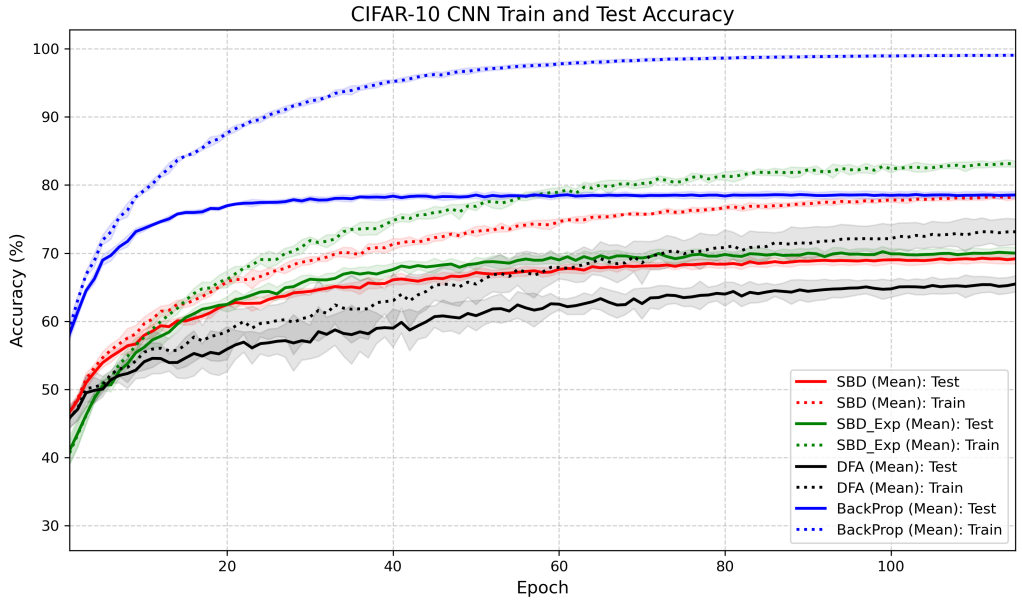


Figure 4: Training and test accuracy curves on CIFAR-10 for BP, DFA, SBD, and SBD with score expansion on the CNN architecture of Section E.2. The expanded-broadcast variant uses the confidence-weighted $\mathbf{p} \odot \boldsymbol{\delta}$ block together with the $\mathrm{roll}_k(\mathbf{p}) \odot \boldsymbol{\delta}$ block, using $k = 5$, from Eq. (7). The displayed interval is shorter than the training budget in Table 7.

E.4 Accuracy curves

Figure 4 shows the CIFAR-10 training and test accuracy curves for BP, DFA, SBD, and the score-expansion variant of SBD on the CNN architecture used in Erdogan et al. (2025) and described in Section E.2. The expansion variant uses the confidence-weighted and `roll(5)` blocks defined in Eq. (7).

The curves are consistent with the final accuracies reported in Table 1. In particular, SBD trained with cross entropy attains higher test accuracy than conventional DFA trained with the same loss, indicating that the score-broadcast update is more effective than the standard direct-feedback alternative in this setting. The score-expansion configuration closely tracks plain SBD early in training and achieves higher accuracy later. This comparison includes the batch-size and learning-rate changes listed above.

Figure 5 shows the corresponding CIFAR-10 training and test accuracy curves for the $4\times$-width CNN of Table 6. In this higher-capacity setting, we report BP and SBD with expanded broadcast vector, where the expansion again uses the confidence-weighted and `roll5` blocks from Eq. (7).

Compared with the baseline-width model, both methods benefit substantially from the $4\times$-width scaling. Using the final accuracies reported in Section 7, BP increases from 78.51% to 83.1%, a gain of 4.59 percentage points, while SBD with expanded broadcast increases from 70.03% to 74.46%, a gain of 4.43 points. The wider configurations preserve the same qualitative ordering. The BP comparison also changes the optimization recipe as described above.

E.5 Multi-seed ablation of score-expansion modulators

The theory of Section 6 and Appendix D permits deterministic X-measurable modulators, provided the required modulated moments exist. We compare several such choices on CIFAR-10, including the $3D$-dimensional expansion used in the main experiments.

Expansion methods. In the formulas below, $\mathbf{a} = \mathbf{a}(X)$ and $\mathbf{p} = \mathbf{p}(X)$ are deterministic functions of the input for fixed network parameters, so each candidate is an X-measurable special case of the general construction.

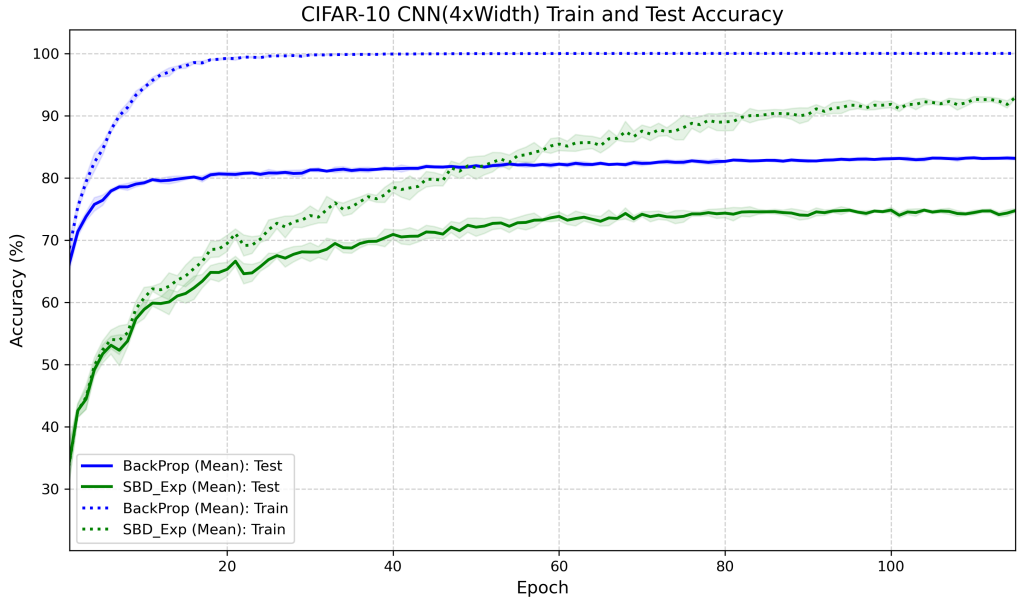


Figure 5: Training and test accuracy curves on CIFAR-10 for BP and SBD with score expansion on the $4\times$-width CNN architecture in Table 6. The SBD variant uses the same expanded broadcast construction as in the baseline setting, namely the confidence-weighted $\mathbf{p} \odot \boldsymbol{\delta}$ block together with the $\mathrm{roll}_k(\mathbf{p}) \odot \boldsymbol{\delta}$ block with $k = 5$. The displayed interval is shorter than the full training budget.

i. `confidence-weighted`

$$\eta_{\text{conf}}(X, Y) \;=\; \mathbf{p}(X) \odot \boldsymbol{\delta} \;\in\; \mathbb{R}^D.$$

The class-d score coordinate is multiplied by the model's predicted probability for class d. Coordinates corresponding to confidently predicted classes contribute proportionally more to the broadcast; coordinates where $p_d \approx 0$ are suppressed.

ii. `logit-weighted`

$$\eta_{\text{logit}}(X, Y) \;=\; \mathbf{a}(X) \odot \boldsymbol{\delta} \;\in\; \mathbb{R}^D.$$

The same idea as `confidence-weighted` but using the raw logits $\mathbf{a}(X)$ instead of the softmax probabilities. Logit magnitudes are unbounded, so this block can produce comparatively large modulators once the network has converged.

iii. `rollk`

$$\eta_{\text{rollk}}(X, Y; k) \;=\; \mathrm{roll}_k(\mathbf{p}(X)) \odot \boldsymbol{\delta} \;\in\; \mathbb{R}^D.$$

A single cyclic shift of $\mathbf{p}(X)$ by k, $k \in \{0, 1, \ldots, D-1\}$, positions provides cross-class coupling through a permutation of the output coordinates. Since class indices are arbitrary, k has no semantic interpretation and does not measure class similarity or distance. The reported experiments simply fix $k = 5$. For $k = 0$ the block is identical to `confidence-weighted`; for $k = 5$ it is identical to `roll5`.

iv. `full-cyclic`

$$\eta_{\text{full}}(X, Y) \;=\; \Big[\mathrm{roll}_1(\mathbf{p}(X)) \odot \boldsymbol{\delta}\,;\; \ldots\;;\; \mathrm{roll}_{D-1}(\mathbf{p}(X)) \odot \boldsymbol{\delta}\Big] \;\in\; \mathbb{R}^{D(D-1)}.$$

Concatenates all $D - 1$ non-trivial cyclic shifts of $\mathbf{p}(X)$ into one block. For $D = 10$ this contributes 90 extra coordinates and captures all binary class-pairings $(d, (d + k) \bmod D)$ for $k = 1, \ldots, 9$ simultaneously. The raw score is also retained, giving 100 broadcast coordinates.

v. `boundary_weight` A band-pass modulator built from two softmax distributions at different temperatures:

$$\phi_{\text{bd}}(X) \;=\; \text{softmax}(\mathbf{a}(X)/T_1) \;-\; \text{softmax}(\mathbf{a}(X)/T_2) \quad \text{with } T_1 = 0.5,\ T_2 = 2.0,$$

$$\eta_{\text{bd}}(X, Y) \;=\; \phi_{\text{bd}}(X) \odot \boldsymbol{\delta} \;\in\; \mathbb{R}^D.$$

This block weights the score by the change in predicted probabilities under temperature scaling. It vanishes for equal logits and in the limit of a single dominant logit; its name does not imply a calibrated measure of distance to a decision boundary.

Evaluation protocol. We evaluated the candidate expansion combinations across multiple independent seeds on the CIFAR-10 CNN of Appendix E. Table 8 reports the mean test accuracy and standard deviation across runs. The selected $3D$-dimensional combination is the same expanded-SBD configuration reported in Table 1, whose result is repeated here for direct comparison.

Modulator-combination ablation. Table 8 reports CIFAR-10 test accuracy for the candidate combinations evaluated. The selected combination $[\boldsymbol{\delta}; \mathbf{p} \odot \boldsymbol{\delta}; \text{roll}_k(\mathbf{p}) \odot \boldsymbol{\delta}]$ with $k = 5$, used in the main text, is highlighted.

Table 8: 5-seed CIFAR-10 top-1 test accuracy for candidate score-expansion modulator combinations. The selected configuration, whose result is also reported in Table 1, is shown in bold.

Expansion	CIFAR-10 (%)
score + boundary-weighted	69.40 ± 0.31
score + confidence-weighted	69.67 ± 0.42
score + full-cyclic	68.96 ± 0.74
score + logit-weighted	61.03 ± 1.89
score + roll5 + confidence-weighted + boundary-weighted	69.44 ± 0.32
score + roll5 + confidence-weighted	$\mathbf{70.0 \pm 0.7}$
score + roll5	68.35 ± 1.46

E.6 Cosine alignment between SBD updates and backpropagation gradients

To quantify the directional relationship between the local SBD direction and the standard backpropagation (BP) gradient, we compute a signed cosine similarity for each hidden layer of the CIFAR-10 CNN during SBD training. Let $S_{e,b}^{(k)}$ denote the unregularized three-factor local SBD direction for layer k at epoch e and mini-batch b, evaluated at the current network parameters before auxiliary regularization and Adam processing. Let $G_{\text{BP},e,b}^{(k)} = \nabla_{W^{(k)}} \mathcal{L}_{\text{CE}}$ denote the BP gradient of the cross entropy loss computed on the same mini-batch and at the same parameter values. The BP gradient is used only as a diagnostic; the network weights are still updated only by the SBD rule. For each hidden layer, both tensors are flattened and compared as

$$c_{e,b}^{(k)} = \frac{\left\langle \text{vec}\left(S_{e,b}^{(k)}\right), \text{vec}\left(G_{\text{BP},e,b}^{(k)}\right) \right\rangle}{\left\| \text{vec}\left(S_{e,b}^{(k)}\right) \right\|_2 \left\| \text{vec}\left(G_{\text{BP},e,b}^{(k)}\right) \right\|_2 + \epsilon_{\text{num}}}, \tag{8}$$

where ϵ_{num} is a small numerical stabilizer. We do not take an absolute value in Eq. (8); hence positive values denote alignment between the local SBD direction and the BP gradient, while negative values denote anti-alignment. A positive cosine makes the negative raw SBD direction a first-order descent direction for the minibatch cross entropy. The corresponding angle, when reported, is $\arccos(\text{clip}(c_{e,b}^{(k)}, -1, 1))$, but Figure 6 plots the cosine similarities themselves. The mini-batch cosine values are averaged within each epoch and the experiment is repeated over five independent seeds. Solid lines in the plot show the across-seed mean, and the shaded envelopes show one standard deviation.

The deepest hidden fully connected layer exhibits the largest cosine similarity, while the convolutional layers have weaker positive alignment.

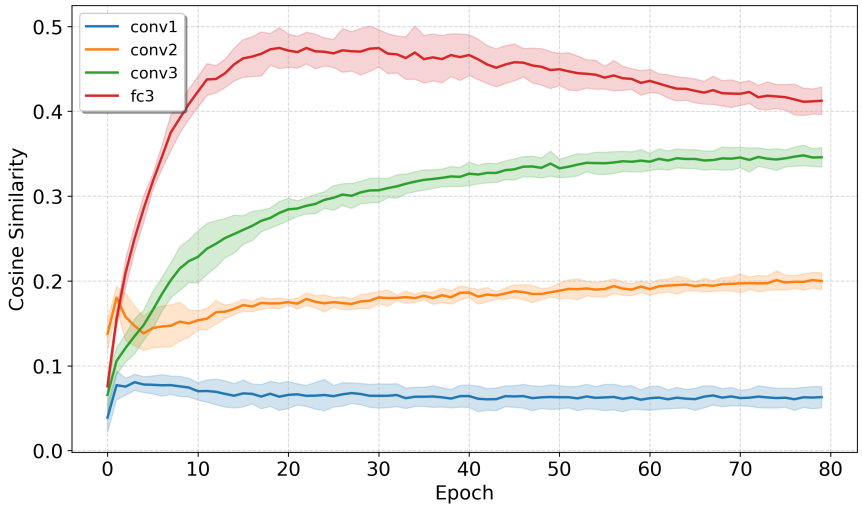


Figure 6: Cosine similarity between the local SBD direction and the diagnostic BP gradient for each hidden layer of the CIFAR-10 CNN. Curves show means over five seeds, and shaded envelopes denote one standard deviation. Mean cosine similarity remains positive in every hidden layer over the plotted interval.

Table 9: CIFAR-10 hyperparameter sensitivity. Entries report test accuracy (%; mean ± standard deviation over five independent seed runs).

λ	0.99993	0.99996	0.99997	0.99998	0.99999	0.999995	0.999999
Test accuracy	10.00 ± 0.00	20.63 ± 4.02	42.70 ± 1.59	59.47 ± 1.99	69.13 ± 0.81	69.33 ± 0.54	67.52 ± 0.72

$c_{\text{cov}}^{(\text{fc4})}$	10^{-5}	10^{-6}	10^{-7}	10^{-8}
Test accuracy	66.96 ± 0.91	69.33 ± 0.64	69.34 ± 0.73	69.26 ± 0.69

Batch size B	1	4	8
Test accuracy	59.36 ± 0.02	66.26 ± 0.31	68.36 ± 0.61

Additional hyperparameter sensitivity. We additionally varied the covariance EMA coefficient λ, the output-layer covariance coefficient $c_{\text{cov}}^{(\text{fc4})}$ (`-CCOV_OUT`), and the minibatch size B, holding all other settings fixed. Performance is stable around the selected EMA regime, $\lambda \in \{0.99999, 0.999995\}$, and across $c_{\text{cov}}^{(\text{fc4})} \in [10^{-8}, 10^{-6}]$; within the tested small-batch regime, accuracy increases with B. Table 9 summarizes these results.

F TINY IMAGENET EXPERIMENTAL SETUP

This appendix documents the codebase used for the Tiny ImageNet experiments reported in the main text, including BP, DFA, plain SBD, and score-expanded SBD. Tiny ImageNet (Le and Yang, 2015) was released as a Stanford CS231N course project, a downsized 200-class subset of ImageNet, without a formal open-source license declaration on its original distribution page, and is used here for non-commercial academic research in accordance with the underlying ImageNet terms of use.

The Tiny ImageNet benchmark consists of 200 classes with 500 training images and 50 validation images per class, all resized to 64×64 RGB. The reported SBD configurations are implemented by per-seed launch scripts in the Tiny ImageNet codebase: plain SBD uses `CNN_tinyImageNet/SBD/run_tinet_sbd_ce_s*.sh`, while score-expanded SBD uses `CNN_tinyImageNet/SBD_exp/run_tinet_sbd_exp_ce_s*.sh`. These scripts call the corresponding Tiny ImageNet trainers, `cnn_tinet.py` for plain SBD and `cnn_tinete.py` for score-expanded SBD, applied to the 6-layer CNN described in Section F.1.

The forward and local update rules follow the CIFAR-10 SBD formulation of Section E.1; this appendix therefore focuses on the Tiny-ImageNet-specific architecture, data pipeline, and hyperparameter choices. All experiments run in `float32` on a single NVIDIA V100 (32GB) GPU. For comparison on a common budget, the approximate times to complete 200 epochs on Tiny ImageNet were 14, 20, 30, and 32 hours for BP, DFA, SBD, and SBD with score expansion, respectively.

Training uses the official training set, and evaluation uses the official validation set because the standard test labels are unavailable. Throughout the Tiny ImageNet results and figures, "test accuracy" refers to accuracy on this official validation set.

For plain SBD, the broadcast vector equals the raw cross-entropy score

$$\boldsymbol{\varepsilon} = \boldsymbol{\delta} = \mathbf{p} - \mathbf{y} \in \mathbb{R}^{200}.$$

For score-expanded SBD, the implementation uses the $3D$-dimensional expansion described in Section E.1,

$$\boldsymbol{\varepsilon} = [\boldsymbol{\delta};\ \mathbf{p} \odot \boldsymbol{\delta};\ \mathrm{roll}_k(\mathbf{p}) \odot \boldsymbol{\delta}] \in \mathbb{R}^{600},$$

selected by `-err_expand=2` in the `SBD_exp` launch scripts, with $k = 5$. The softmax temperature and broadcast scale are fixed at $T = 1$ and $\alpha = 1$ in the Tiny ImageNet implementation.

F.1 Network architecture

The Tiny ImageNet network is a wider and deeper variant of the CIFAR-10 backbone. It uses three 5×5 convolutional blocks with max-pooling, followed by three fully connected layers with ReLU and dropout. Every convolution and linear layer contains a bias, but the implemented SBD variants leave all biases at their zero initialization; DFA updates only FC biases, and BP updates all biases. A single `-width_multiplier` flag scales all conv channels and the FC hidden width jointly.

Table 10: Tiny ImageNet CNN architecture at `-width_multiplier`$= 1.0$. Input: Tiny ImageNet images of shape $(3,\ 64,\ 64)$. Total parameters, including biases: 14,130,888.

Layer	Operation	Output shape	Parameters
conv1	5×5 conv, stride 2, pad 2	$(96,\ 32,\ 32)$	7,296
act1	ReLU	$(96,\ 32,\ 32)$	–
pool1	MaxPool 2×2	$(96,\ 16,\ 16)$	–
conv2	5×5 conv, stride 1, pad 2	$(128,\ 16,\ 16)$	307,328
act2	ReLU	$(128,\ 16,\ 16)$	–
pool2	MaxPool 2×2	$(128,\ 8,\ 8)$	–
conv3	5×5 conv, stride 1, pad 2	$(256,\ 8,\ 8)$	819,456
act3	ReLU	$(256,\ 8,\ 8)$	–
pool3	MaxPool 2×2	$(256,\ 4,\ 4)$	–
flatten	reshape	$(4096,)$	–
fc3	linear	$(2048,)$	8,390,656
act4	ReLU + Dropout(p)	$(2048,)$	–
fc4	linear	$(2048,)$	4,196,352
act5	ReLU + Dropout(p)	$(2048,)$	–
fc5	linear	$(200,)$	409,800
softmax	softmax at $T=1$	$(200,)$	–

The conv channels P_0, P_1, P_2 and FC hidden width F_h are parameterized as

$$P_0 = \mathrm{round}(96\cdot w), \quad P_1 = \mathrm{round}(128\cdot w), \quad P_2 = \mathrm{round}(256\cdot w), \quad F_h = \mathrm{round}(2048\cdot w),$$

where $w =$ `width_multiplier`. The fc3 input dimension $P_2 \cdot 4 \cdot 4$ is derived automatically from the conv stack, giving 4096 at `-width_multiplier`$= 1.0$. Weights are initialized from a Gaussian distribution with standard deviation $\sqrt{2/(\texttt{scale_factor} \cdot \text{fan_in})}$, giving a scaled variant of He et al.'s fan-in initialization (He et al., 2015) with `scale_factor` $= 6$; biases are initialized to zero.

F.2 Data pipeline and augmentation

All images are normalized per-channel using the standard ImageNet statistics

$$\boldsymbol{\mu} = (0.485,\ 0.456,\ 0.406), \qquad \boldsymbol{\sigma} = (0.229,\ 0.224,\ 0.225).$$

Training augmentation consists of (i) a random crop to 64×64 from a 4-pixel reflection-padded image and (ii) random horizontal flipping, applied independently per sample and per epoch. No color jittering, mixup, or RandAugment is used in the main-text configuration.

For CE BP, DFA, SBD, and expanded SBD, training and validation accuracies use the same deterministic mirror-and-translation ensemble implemented in `infer_mirror_translate`. For each image, the code first averages the predicted probabilities over the original image and its horizontal flip. It then applies two deterministic reflection-padded translation shifts and averages the mirror-averaged outputs of these translated views. The final prediction is

$$0.5\,\bar{\mathbf{y}}_{\mathrm{mirror}} + 0.5\,\bar{\mathbf{y}}_{\mathrm{trans}},$$

where $\bar{\mathbf{y}}_{\mathrm{mirror}}$ is the mean of two forward passes and $\bar{\mathbf{y}}_{\mathrm{trans}}$ is the mean over two translated mirror pairs. The evaluation data loader itself applies no stochastic augmentation beyond normalization.

Evaluation disables dropout and runs after every training pass.

F.3 Training hyperparameters

Table 11 lists the training hyperparameters used by the Tiny ImageNet score-expanded SBD launch scripts `CNN_tinyImageNet/SBD_exp/run_tinet_sbd_exp_ce_s*.sh`. The ***SBD coefficients*** block follows the layer indexing of Section E.1, with `CMSE_OUT` now applied to the deeper output layer fc5, `CMSE_OUT2` shared between the two hidden FC layers fc3 and fc4, and `CMSE_HIDDEN` shared across all three convolutional layers. The covariance coefficients use the analogous grouping, while `CL1_OUT` is applied to the two hidden FC layers fc3 and fc4 in the Tiny ImageNet implementation. Each of these layers receives both the activation ℓ_1 term of Section E.1 and an additional weight ℓ_1 term $c_{\ell_1}^{(k)}\mathrm{sign}(W^{(k)})$. The local FC rules use post-dropout activations in the moment estimates and the ReLU gate in the score and covariance directions.

Plain SBD shares the expanded-SBD optimization settings. The CE BP and DFA runs use initial learning rates 10^{-4} and 5×10^{-5}, respectively, and dropout probabilities 0.10 and 0.05, compared with 0.08 for both SBD variants. All four use batch size 28, Adam weight decay 1.25×10^{-5}, and learning-rate decay 0.98. The SBD training budget comprises 251 passes, indexed 0–250; incomplete training batches are dropped. Learning-rate and EMA schedules start at zero-based epoch 2, as in CIFAR-10.

The Tiny ImageNet expanded-SBD hyperparameters in Table 11 were selected by a grid search over the SBD learning-rate and regularization coefficients. The search was managed using Weights & Biases sweeps (Biewald, 2020), including automatic early termination of clearly underperforming configurations.

F.4 Accuracy curves

Figure 8 and Figure 7 show the Tiny ImageNet test and training accuracy curves for BP, SBD, and SBD with score expansion. The curves show a clear qualitative separation between the methods. BP obtains the highest test accuracy, reaching 39.89%, while SBD with score expansion reaches 31.43% and plain SBD reaches 28.29%. Thus, as in the CIFAR-10 experiments, expanding the broadcast vector improves the SBD update, yielding a gain of 3.14 percentage points over using the score vector alone.

Figure 9 shows the corresponding DFA training and test accuracy curves. DFA improves slowly over the longer 500-epoch run, reaching 17.49% test accuracy and 21.02% training accuracy at the end of training. Even with this longer training horizon, DFA remains well below the SBD variants, indicating that the score-broadcast update is more effective than the standard direct-feedback alternative on the Tiny ImageNet CNN experiment as well.

Convergence speed and measured GPU energy. Table 12 reports the median first-passage epoch and cumulative GPU energy required to reach fixed evaluation-accuracy targets. Because the methods converge to different final accuracies, we also include a method-relative target: writing $A_{\mathcal{M}}^{\mathrm{final}}$ for the final evaluation accuracy of method $\mathcal{M}$, this threshold is $0.95A_{\mathcal{M}}^{\mathrm{final}}$. All runs used a single NVIDIA V100 (32GB), and the energy values are GPU-only cumulative measurements from the system logs. Each table entry is epoch/kWh; NR denotes that the target was not reached.

Table 11: Tiny ImageNet expanded-SBD training hyperparameters. Values shown correspond to the score-expanded SBD CE launch scripts `CNN_tinyImageNet/SBD_exp/run_tinet_sbd_exp_ce_s*.sh`. Column *Flag* is the CLI name in the launch script; column *Symbol* names the variable used in the equations of Section E.1.

Group	Flag	Symbol	Value
General	`-logger_name`	seed	5 independent seeds
	`-n_epochs`	–	251
	`-batch_size`	B	28
	`-loss_out`	–	cross entropy (`ce`)
	augmentation	–	RandomCrop pad 4 + HFlip
Method	`-method`	–	`sbd`
	`-err_expand`	–	2
	broadcast vector	$\boldsymbol{\varepsilon}$	$3D$-dimensional expansion of $\boldsymbol{\delta}$
	softmax temperature	T	1.0 fixed
	effective D_ε	D_ε	600
Optimizer	`-lr`	η_0	8.5×10^{-5}
	`-lr_drop_rate`	–	0.98
	`-lr_drop_every`	–	1 epoch
	`-weight_decay`	–	1.25×10^{-5}
	Adam β_1, β_2	–	0.9, 0.999
	precision	–	`float32`
SBD coefficients	`-CMSE_OUT`	$c_{\text{score}}^{(\text{fc5})}$	10
	`-CMSE_OUT2`	$c_{\text{score}}^{(\text{fc3,fc4})}$	0.1
	`-CMSE_HIDDEN`	$c_{\text{score}}^{(\text{conv})}$	0.1
	`-CCOV_OUT`	$c_{\text{cov}}^{(\text{fc5})}$	1×10^{-7}
	`-CCOV_OUT2`	$c_{\text{cov}}^{(\text{fc3,fc4})}$	6×10^{-7}
	`-CCOV_HIDDEN`	$c_{\text{cov}}^{(\text{conv})}$	0
	`-CL1_OUT`	$c_{\ell_1}^{(\text{fc3,fc4})}$	1×10^{-7}
	`-CL1_HIDDEN`	$c_{\ell_1}^{(\text{hidden})}$	0
Covariance EMA	`-Reh_lambda`	λ	0.999992
	`-Reh_lambda2`	λ_2	0.999992
	`-Reh_lambda_drop`	ρ	0.015
	`-Reh_lambda_drop_every`	–	1 epoch
	`-Reh_gain`	$\text{std}(\widehat{\mathbf{R}}_{g\varepsilon}^{(\text{conv})})$	0.01
	`-Reh_gain_lin`	$\text{std}(\widehat{\mathbf{R}}_{g\varepsilon}^{(\text{FC})})$	0.01
	`-Reh_ini`	init diagonal of $\widehat{\mathbf{R}}_{hh}$	1×10^{-8}
Architecture	`-width_multiplier`	w	1.0
	`-dropout`	p	0.08
	`-scale_factor`, `-init_dist`	init scale, dist.	6, Gaussian

BP reaches these targets fastest and with the lowest measured energy in the present implementation. This comparison should not be interpreted as an intrinsic hardware-efficiency advantage: BP uses highly optimized PyTorch/CUDA kernels, whereas the SBD variants use custom layer-local update code. Among the broadcast methods, expanded SBD reaches each attainable fixed target before plain SBD, is the only method other than BP to reach 30%, and reaches its method-relative target before both plain SBD and DFA. Both SBD variants require substantially fewer epochs and less measured energy than DFA at the targets they share.

A notable difference from the SBD and DFA curves is the strong overfitting observed for BP. In the training plot, BP rapidly approaches nearly perfect training accuracy, reaching 99.98%, whereas its test accuracy saturates at 39.89%. This corresponds to a train-test gap of 60.09 percentage points. In contrast, the SBD variants achieve lower training accuracy and also lower test accuracy, but exhibit a

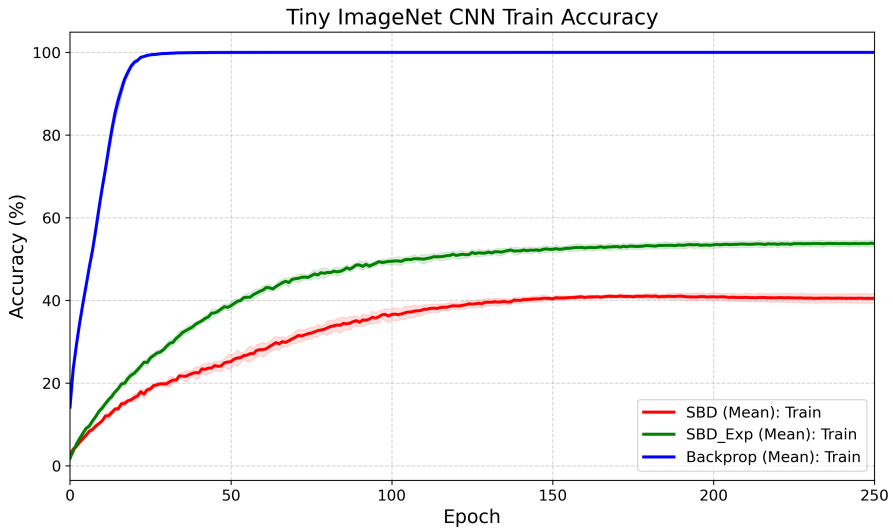


Figure 7: Training accuracy on Tiny ImageNet for BP, SBD, and SBD with score expansion (SBD_Exp), shown at epochs 0–250. Each curve represents the mean over five runs, and the shaded region denotes one standard deviation.

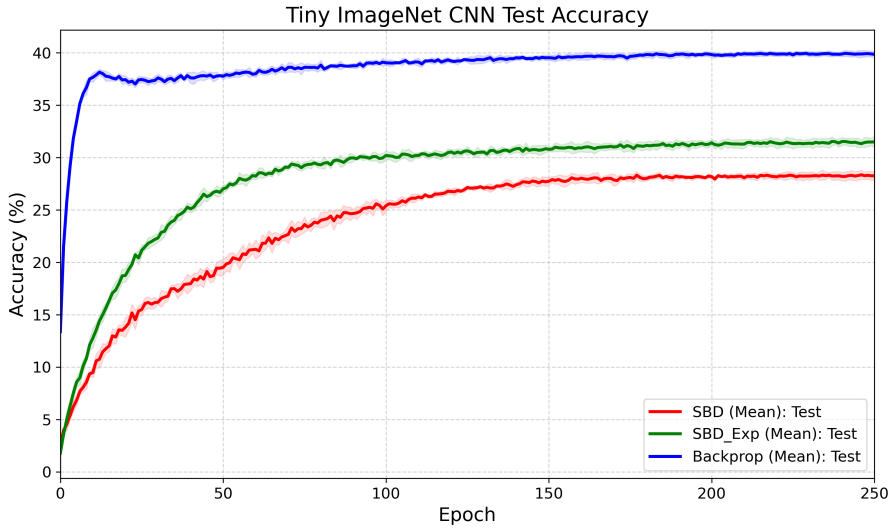


Figure 8: Test accuracy on Tiny ImageNet for BP, SBD, and SBD with score expansion (SBD_Exp), shown at epochs 0–250. Each curve represents the mean over five runs, and the shaded region denotes one standard deviation.

much less extreme separation between training and test performance. The expanded SBD model, for example, reaches 53.77% training accuracy and 31.43% test accuracy. This suggests that BP fits the Tiny ImageNet training set very aggressively in this architecture, while the broadcast-based learning rules act as a more constrained optimization procedure.

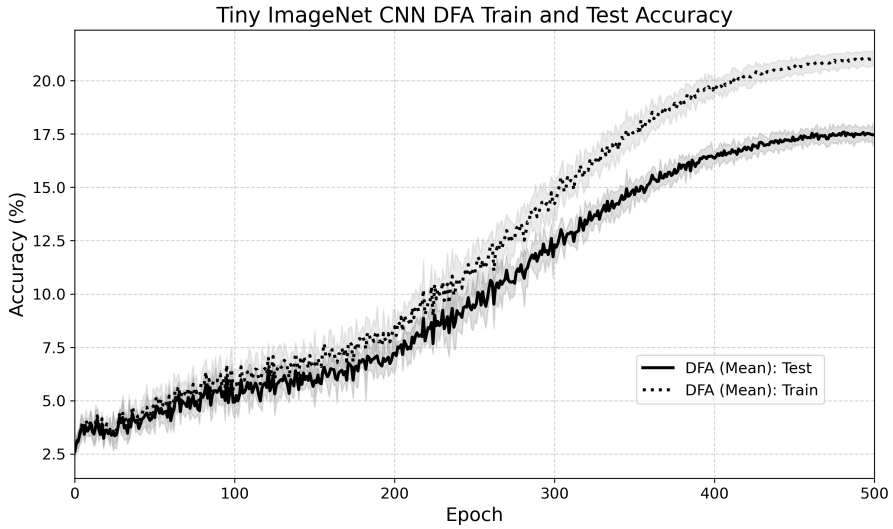


Figure 9: Training and test accuracy on Tiny ImageNet for DFA across 500 epochs. Each curve represents the mean over five runs, and the shaded region denotes one standard deviation. DFA improves slowly throughout training and remains substantially below the accuracies obtained by BackProp and the SBD variants.

Table 12: Tiny ImageNet convergence and measured GPU energy. Entries give the median first-passage epoch and cumulative GPU-only energy in kWh.

Evaluation target	BP	SBD	SBD Exp.	DFA
15%	4/0.028	26/0.447	16/0.358	363/3.640
25%	5/0.033	95/1.497	42/0.913	NR
30%	6/0.037	NR	95/1.968	NR
95% of method-specific final accuracy	46/0.218	129/2.093	93/1.991	410/4.118

G Correlative Information Maximization (CorInfoMax) Network Experiments

This appendix describes the CorInfoMax-based network used in our additional experiments. The construction combines two previously separate ideas. The internal representation dynamics, asymmetric predictors, and lateral interactions follow the supervised Correlative Information Maximization (CorInfoMax) framework of Bozkurt et al. (2023a). Supervised plasticity, however, is supplied by the cross-entropy score developed in this paper rather than by the two-state equilibrium-propagation procedure used in the original CorInfoMax formulation. This distinction is also important relative to the CorInfoMax-EBD model of Erdogan et al. (2025): CorInfoMax-EBD broadcasts an MSE output error, whereas the present CorInfoMax-SBD models use cross-entropy-based supervision. We first describe the underlying construction, then specify the score, connectivity, and numerical controls used in each experiment.

G.1 CorInfoMax background and score-broadcast learning

The following background follows Bozkurt et al. (2023a); the experimental variants below disable the top-down predictor pathway.

G.1.1 Bidirectional prediction view of CorInfoMax

Let $\mathbf{h}^{(0)} = \mathbf{x}$ and let $\mathbf{h}^{(k)} \in \mathbb{R}^{N^{(k)}}$, $k = 1, \ldots, L$, denote the states of a recurrent hierarchy. Each adjacent pair is connected by two independently parameterized linear predictors. The feedforward matrix $\mathbf{W}_{\mathrm{ff}}^{(k)}$ predicts layer $k+1$ from layer k, while $\mathbf{W}_{\mathrm{fb}}^{(k)}$ predicts layer k from layer $k+1$. Their

instantaneous residuals are

$$\overrightarrow{\mathbf{e}}^{(k+1)} = \mathbf{h}^{(k+1)} - \mathbf{W}_{\text{ff}}^{(k)}\mathbf{h}^{(k)}, \tag{9}$$

$$\overleftarrow{\mathbf{e}}^{(k)} = \mathbf{h}^{(k)} - \mathbf{W}_{\text{fb}}^{(k)}\mathbf{h}^{(k+1)}. \tag{10}$$

No transpose constraint is imposed between $\mathbf{W}_{\text{ff}}^{(k)}$ and $\mathbf{W}_{\text{fb}}^{(k)}$.

For a minibatch indexed by m, write $\mathbf{H}^{(k)}[m] \in \mathbb{R}^{N^{(k)} \times B}$ for the layer-state matrix and use analogous notation for the two residual matrices. In this background formulation, second-order statistics can be tracked by exponential averages; for example,

$$\widehat{\mathbf{R}}_h^{(k)}[m] = \lambda_h \widehat{\mathbf{R}}_h^{(k)}[m-1] + \frac{1-\lambda_h}{B}\mathbf{H}^{(k)}[m]\mathbf{H}^{(k)}[m]^T, \tag{11}$$

$$\widehat{\mathbf{R}}_{\to e}^{(k+1)}[m] = \lambda_h \widehat{\mathbf{R}}_{\to e}^{(k+1)}[m-1] + \frac{1-\lambda_h}{B}\overrightarrow{\mathbf{E}}^{(k+1)}[m]\overrightarrow{\mathbf{E}}^{(k+1)}[m]^T, \tag{12}$$

with a corresponding recursion for $\widehat{\mathbf{R}}_{\leftarrow e}^{(k)}$.

Correlative mutual information between two neighboring states admits a forward-prediction form and a backward-prediction form:

$$\widehat{I}_{\to}^{(\epsilon_k)}(\mathbf{h}^{(k)}, \mathbf{h}^{(k+1)}) = \frac{1}{2}\log\det\left(\widehat{\mathbf{R}}_h^{(k+1)} + \epsilon_k\mathbf{I}\right) - \frac{1}{2}\log\det\left(\widehat{\mathbf{R}}_{\to e}^{(k+1)} + \epsilon_k\mathbf{I}\right), \tag{13}$$

$$\widehat{I}_{\leftarrow}^{(\epsilon_k)}(\mathbf{h}^{(k)}, \mathbf{h}^{(k+1)}) = \frac{1}{2}\log\det\left(\widehat{\mathbf{R}}_h^{(k)} + \epsilon_k\mathbf{I}\right) - \frac{1}{2}\log\det\left(\widehat{\mathbf{R}}_{\leftarrow e}^{(k)} + \epsilon_k\mathbf{I}\right). \tag{14}$$

The first log-determinant in each expression rewards a representation that uses the available dimensions, while the second penalizes variation that is not linearly predictable from the neighboring layer. For an interior layer, the state-dependent CorInfoMax criterion is chosen as

$$\mathcal{C}^{(k)} = \widehat{I}_{\to}^{(\epsilon_{k-1})}(\mathbf{h}^{(k-1)}, \mathbf{h}^{(k)}) + \widehat{I}_{\leftarrow}^{(\epsilon_k)}(\mathbf{h}^{(k)}, \mathbf{h}^{(k+1)}). \tag{15}$$

Using opposite prediction directions for the two adjacent terms yields separate bottom-up and top-down pathways without requiring weight transport.

G.1.2 Free-state recurrent inference

CorInfoMax inference updates all non-input states while the predictor and lateral parameters are fixed. Let $\mathbf{u}^{(k)}$ be the somatic variable before projection and let

$$\mathbf{C}_{\text{ideal}}^{(k)}[m] = \left(\widehat{\mathbf{R}}_h^{(k)}[m] + \epsilon_k\mathbf{I}\right)^{-1} \tag{16}$$

denote the inverse-correlation operator motivating lateral adaptation. The implemented lateral matrix $\mathbf{C}^{(k)}$ is updated recursively, rather than recomputed by this inversion. A background form of the hidden-state dynamics is

$$\tau_u\frac{d\mathbf{u}^{(k)}}{ds} = -g_{\text{leak},k}\mathbf{u}^{(k)} + 2\gamma_h\mathbf{C}^{(k)}\mathbf{h}^{(k)} - \frac{1}{\epsilon_{k-1}}\overrightarrow{\mathbf{e}}^{(k)} - \frac{1}{\epsilon_k}\overleftarrow{\mathbf{e}}^{(k)}, \tag{17}$$

$$\mathbf{h}^{(k)} = \Pi_{\mathcal{P}^{(k)}}\left(\mathbf{u}^{(k)}\right), \tag{18}$$

where s indexes the recurrent relaxation, γ_h depends on the correlation forgetting factor, and $\Pi_{\mathcal{P}^{(k)}}$ enforces the chosen activity domain. In our experiments the hidden projection is an elementwise ReLU clipped to $[0, 50]$; the output state is unbounded. The top-down drive is disabled, while basal and lateral interactions remain active. These choices specialize the general CorInfoMax dynamics.

Equation (17) can be read as a multi-compartment neuron model. The drive $\mathbf{W}_{\text{ff}}^{(k-1)}\mathbf{h}^{(k-1)}$ enters through a basal pathway, whereas $\mathbf{W}_{\text{fb}}^{(k)}\mathbf{h}^{(k+1)}$ and the lateral signal $\mathbf{C}^{(k)}\mathbf{h}^{(k)}$ contribute through an apical pathway. The somatic state integrates these terms subject to leakage and the activity constraint. Under this interpretation, the two prediction errors are voltage mismatches between the soma and its basal or apical prediction. Off-diagonal entries of the lateral operator can be implemented through

inhibitory interneurons, while its diagonal component acts as a self-interaction. These architectural interpretations are inherited from CorInfoMax.

Unlike CorInfoMax-EP, our training pass does not introduce a second, label-nudged phase. The recurrent system performs a fixed number of label-free iterations: fifteen for the fully connected model and three for the locally connected model. The final states need not be equilibria. The last layer is then interpreted as logits $\mathbf{a} = \mathbf{h}^{(L)}$ and used to form the supervised score.

G.1.3 Cross-entropy score-broadcast learning

For one-hot labels $\mathbf{y}$, the classifier output and its score are

$$\mathbf{p} = \mathrm{softmax}(\mathbf{a}), \qquad \boldsymbol{\delta}_{\mathrm{CE}} = \nabla_{\mathbf{a}} \mathcal{L}_{\mathrm{CE}} = \mathbf{p} - \mathbf{y}. \tag{19}$$

This is the unregularized, unsmoothed score used in the locally connected experiment. The fully connected experiment modifies it as specified below. For the base construction, on minibatch m, let $\boldsymbol{\Delta}_{\mathrm{CE}}[m] \in \mathbb{R}^{D \times B}$ collect these scores. For every hidden layer, define a feature transform $g^{(k)}$ and the matrix

$$\mathbf{G}^{(k)}[m] = \left[g^{(k)}(\mathbf{h}^{(k)}[mB+1]), \ldots, g^{(k)}(\mathbf{h}^{(k)}[mB+B])\right]. \tag{20}$$

The adaptive score-projection matrix is updated by

$$\widehat{\mathbf{R}}^{(k)}_{g\delta}[m] = \lambda_d \widehat{\mathbf{R}}^{(k)}_{g\delta}[m-1] + \frac{1-\lambda_d}{B} \mathbf{G}^{(k)}[m] \boldsymbol{\Delta}_{\mathrm{CE}}[m]^T, \tag{21}$$

and converts the global score into a layer-width modulation,

$$\mathbf{Q}^{(k)}[m] = \widehat{\mathbf{R}}^{(k)}_{g\delta}[m] \boldsymbol{\Delta}_{\mathrm{CE}}[m]. \tag{22}$$

If $\mathbf{F}^{(k)}_d$ contains the local gates for the state projection and $\mathbf{G}^{(k)}_d$ contains the elementwise derivatives of $g^{(k)}$, the local postsynaptic factor used by plasticity is

$$\boldsymbol{\Phi}^{(k)}[m] = \mathbf{F}^{(k)}_d[m] \odot \mathbf{G}^{(k)}_d[m] \odot \mathbf{Q}^{(k)}[m]. \tag{23}$$

This is the recurrent-network counterpart of the SBD three-factor term in the main paper: a presynaptic state is multiplied by a postsynaptic sensitivity and a projected, loss-derived modulatory signal. Score expansion can be used without changing the construction by replacing $\boldsymbol{\Delta}_{\mathrm{CE}}$ with the expanded score matrix of Appendix D.2. Both experiments use $g^{(k)}(\mathbf{h}) = \mathbf{h}$ and uncentered hidden states in these correlations. Local updates hold inference states and broadcast signals fixed.

G.1.4 Predictor and lateral plasticity

The CorInfoMax and SBD contributions are combined at the synapses rather than in the state objective. For an interior layer k, a convenient minibatch form of the incoming feedforward update is

$$\Delta \mathbf{W}^{(k-1)}_{\mathrm{ff}} = \frac{\mu_{\mathrm{ff},k}}{B} \overrightarrow{\mathbf{E}}^{(k)} \mathbf{H}^{(k-1)T} - \frac{\mu_{\mathrm{SBD,ff},k}}{B} \boldsymbol{\Phi}^{(k)} \mathbf{H}^{(k-1)T}, \tag{24}$$

while the top-down predictor is updated by

$$\Delta \mathbf{W}^{(k)}_{\mathrm{fb}} = \frac{\mu_{\mathrm{fb},k}}{B} \overleftarrow{\mathbf{E}}^{(k)} \mathbf{H}^{(k+1)T} - \frac{\mu_{\mathrm{SBD,fb},k}}{B} \boldsymbol{\Phi}^{(k)} \mathbf{H}^{(k+1)T}. \tag{25}$$

The first term in each equation improves prediction between neighboring layers. The second term is the cross-entropy SBD correction and has the same three-factor structure for both pathways. Because the two matrices receive their own residuals and have independent learning rates, these rules do not create a hidden transpose relation between the feedforward and feedback connections.

The lateral operator simultaneously tracks layer statistics and participates in score-driven plasticity. With $\mathbf{Z}^{(k)} = \mathbf{C}^{(k)} \mathbf{H}^{(k)}$, its update can be written schematically as

$$\begin{aligned} \mathbf{C}^{(k)}[m] = \frac{1}{\lambda_C} \mathbf{C}^{(k)}[m-1] - \frac{\gamma_C}{B} \mathbf{Z}^{(k)}[m] \mathbf{Z}^{(k)}[m]^T \\ - \frac{\mu_{\mathrm{SBD,lat},k}}{B} \left(\boldsymbol{\Phi}^{(k)}[m] \mathbf{H}^{(k)}[m]^T + \mathbf{H}^{(k)}[m] \boldsymbol{\Phi}^{(k)}[m]^T \right). \end{aligned} \tag{26}$$

The first two terms are motivated by online inverse-correlation adaptation; the last is the symmetric score-driven correction. The latter is disabled in the fully connected experiment and enabled in the locally connected experiment, where all lateral products are restricted to the specified local connections.

In the base construction, the output uses the unexpanded score. Equivalently, one may set the score projection to the identity and $g^{(L)}$ to the identity, so that the supervised component of the incoming-weight update reduces to the usual outer product $-B^{-1}\mathbf{\Delta}_{\mathrm{CE}}\mathbf{H}^{(L-1)T}$.

G.1.5 Single-phase training cycle and relation to prior variants

For each minibatch, CorInfoMax-SBD performs the following sequence:

1. Initialize the recurrent states and run the fixed number of free-state iterations using the current active pathways.
2. Read the final-layer logits, evaluate $\mathbf{p} = \mathrm{softmax}(\mathbf{a})$, and form the experiment's score.
3. Update each layer's score correlation $\widehat{\mathbf{R}}_{g\delta}^{(k)}$, then compute $\mathbf{Q}^{(k)}$ and $\mathbf{\Phi}^{(k)}$.
4. Apply the neighboring-layer prediction updates together with the SBD terms in Equations (24)– (26) on active pathways; update the output weights from the unexpanded score.

In both experiments the neighboring-layer prediction-plasticity coefficient is 10^{-18}, making its contribution negligible. Training also uses activity sparsity and power control, momentum, and periodic bounds on feedforward, broadcast, and lateral weight norms. An activity-power controller reduces supervised update rates and damps momentum when activity exceeds the layer thresholds. The power-control update includes a surrogate gate at the upper clipping bound so that saturated units can recover. These numerical controls are part of the experimental recipe; their exact settings are provided in the accompanying launch scripts.

Table 13 summarizes the methodological boundary between the three related models, particularly the source and timing of supervision.

Table 13: Supervision mechanisms in related CorInfoMax training schemes.

Method	Supervised signal	State relaxations	Synaptic credit signal
CorInfoMax-EP (Bozkurt et al., 2023a)	MSE label nudge	Free and nudged	Difference of local statistics between equilibria
CorInfoMax-EBD (Erdogan et al., 2025)	MSE output error	Free only	Projected error–feature correlation
CorInfoMax-SBD (ours)	CE-based score	Free only	Projected score–feature correlation

The experimental models retain CorInfoMax-inspired basal and lateral dynamics and use single-phase score-broadcast plasticity, without a label-nudged inference pass.

G.2 The 10-stage fully connected CorInfoMax experiment

G.2.1 Architecture of the 10-stage fully-connected model

The 10-stage fully connected model has layer dimensions

$$[3072,\ 4000,\ 500,\ 500,\ 500,\ 500,\ 500,\ 500,\ 500,\ 500,\ 10]. \tag{27}$$

The input is a flattened CIFAR-10 image with $3 \times 32 \times 32 = 3072$ coordinates. The network has a 4000-unit first hidden state, eight successive 500-unit hidden states, and a ten-unit classifier state. Thus the architecture vector has eleven state dimensions when the input is included, and ten weight-bearing transformations:

$$3072 \to 4000 \to \underbrace{500 \to \cdots \to 500}_{8 \text{ hidden states}} \to 10. \tag{28}$$

The hidden states are assigned the spatial shapes

$$3072 = 3 \times 32 \times 32, \qquad 4000 = 10 \times 20 \times 20, \qquad 500 = 5 \times 10 \times 10. \tag{29}$$

These shapes are used only to construct the local initialization masks; the model stores the states as vectors.

The feedforward matrices have the standard CorInfoMax orientation

$$\mathbf{W}_{\mathrm{ff}}^{(k)} \in \mathbb{R}^{N^{(k+1)} \times N^{(k)}}, \qquad k = 0, \ldots, 9, \tag{30}$$

and the model also allocates independently parameterized feedback predictors $\mathbf{W}_{\mathrm{fb}}^{(k)}$ and lateral CorInfoMax operators $\mathbf{B}^{(k)}$ (the $\mathbf{C}^{(k)}$ operators above). Score-broadcast matrices $\mathbf{W}_{\mathrm{eb}}$ implement the adaptive score correlations. The three-block expansion gives a hidden broadcast dimension of 30.

Score used in the fully connected experiment. With $\bar{a} = 10^{-1} \sum_j a_j$, the output score is

$$\boldsymbol{\delta}_{\mathrm{FC}} = \mathbf{p} - (0.97\mathbf{y} + 0.03\mathbf{1}_{10}/10) + \frac{0.01}{10}(\mathbf{a} - \bar{a}\mathbf{1}_{10}).$$

This is the derivative of label-smoothed CE plus the centered-logit penalty $0.01\|\mathbf{a} - \bar{a}\mathbf{1}_{10}\|_2^2/(2 \cdot 10)$. The hidden broadcast concatenates $[\boldsymbol{\delta}_{\mathrm{FC}}; \mathbf{p} \odot \boldsymbol{\delta}_{\mathrm{FC}}; \mathrm{roll}_5(\mathbf{p}) \odot \boldsymbol{\delta}_{\mathrm{FC}}]$ and subtracts its running mean, updated with the same forgetting factor as the score correlations. The output uses the uncentered, unexpanded score. The additional logit penalty and running-mean subtraction are experimental modifications; the conditional mean-zero theorem for the raw CE score does not directly establish orthogonality for this modified broadcast.

Sparse local initialization. Local connectivity is enabled for feedforward stages 0 through 8, with kernel sizes

$$(11,\ 7,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5). \tag{31}$$

The final $500 \to 10$ classifier, stage 9, remains dense. For each masked stage, the implementation aligns the source and destination spatial grids, retains only the corresponding local receptive-field entries, and initializes the allowed edges using a mask-aware Xavier scale. The local pattern sparsifies the initial weights but is not reimposed after local plasticity updates. This is local structural initialization, not a permanently sparse LC network.

Dynamics and connection pathways. The basal pathway is the feedforward predictor, the apical pathway is the feedback predictor, and the lateral pathway is the adaptive CorInfoMax operator. The active experiment uses the feedforward, lateral, and CE broadcast pathways, and does not use feedback weights. Each batch-1 sample is processed with fifteen free neural-dynamics iterations, followed by local CorInfoMax/SBD updates. The fully connected experiment additionally uses an Oja-type activity-dependent weight shrinkage. The implementation performs no backpropagation and does not use an autograd optimizer.

G.2.2 Training settings

Table 14 summarizes the settings for the 10-layer fully-connected CorInfoMax experiments.

Table 14: Hyperparameters for 10-layer fully connected CorInfoMax network.

Quantity	Value
Dataset and batch	CIFAR-10; batch size $B = 1$
Architecture	$[3072, 4000, 500, 500, 500, 500, 500, 500, 500, 500, 10]$
Trainable stages	10 feedforward stages; 9 hidden states and 1 classifier
Objective	Smoothed CE plus centered-logit penalty; three-block expansion
CE temperature	1.0
Label smoothing	0.03
Free neural iterations	15 per sample update
Feedback predictor pathway	Disabled
Broadcast weight ($\mathbf{W}_{eb}$) gain	1.0 Xavier gain
Supervised update reference coefficients	$(3.5, 3.5, 3.5, 3.5, 3.5, 3.5, 1.75, 1.75, 1.75, 60000)$
Lateral forgetting	$\lambda = 0.999999949999981$
Broadcast weight forgetting	$\lambda_{\text{eb}} : 0.999999687499268 \rightarrow 0.999999937499971$
Schedule	Coupled cosine; warmup 1, decay $20 \rightarrow 100$ epochs, floor 0.50
Direct/output scales	Output error scale 0.0025; lateral output scale 0.07; broadcast scale 1.0
Local initialization	Enabled on feedforward stages 0–8; final classifier dense
Persistent mask	Disabled; local sparsity is initialization-only
Augmentation	Enabled from epoch 8; reflect padding, crop padding 2, horizontal flip
Activity controls	Scale 0.15; power ceilings $(1, 12, 14, 14, 14, 14, 8, 4, 1.25, 5)$

Figure 10 shows the train and test accuracy trajectories for the 10-layer fully connected CorInfoMax-SBD model. Both curves rise rapidly during the first epochs and pass approximately 50% accuracy early in training. Test accuracy subsequently improves more gradually and approaches 60%, whereas training accuracy continues to increase to approximately 71%. The standard-deviation envelopes are widest during the early transition and become narrower later in training; the increasing train–test separation indicates a growing generalization gap after the initial rapid improvement.

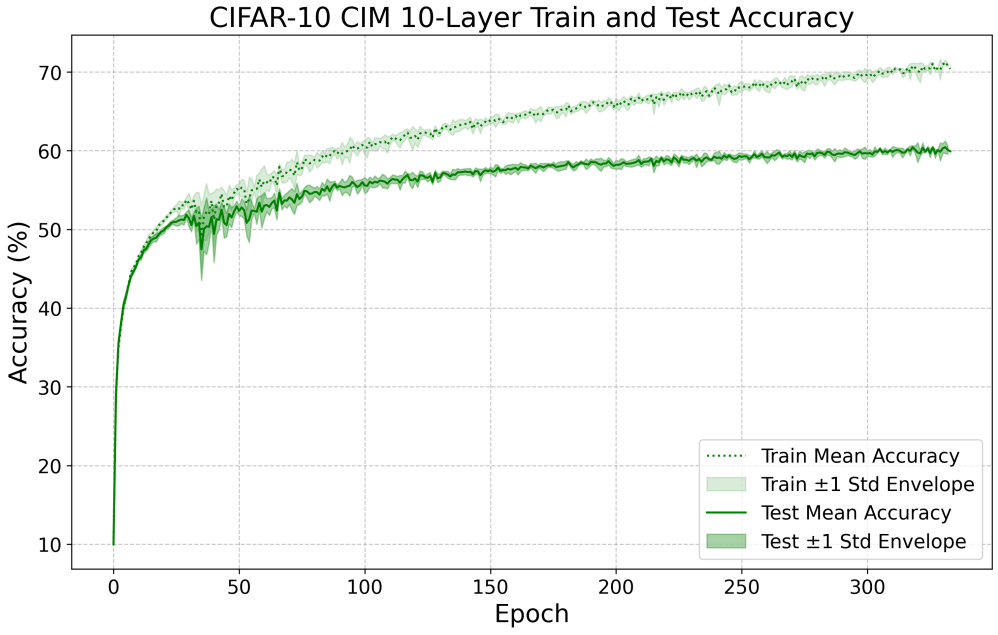


Figure 10: Training and test accuracy of the 10-layer fully connected CorInfoMax-SBD model on CIFAR-10. Curves show the mean over five independent seed runs, and shaded regions indicate one standard deviation. The dotted and solid curves denote training and test accuracy, respectively.

G.3 Locally Connected CorInfoMax Experiments

Convolutional neural networks (CNNs) obtain much of their efficiency and visual inductive bias by applying the same learned filters at every spatial location. This exact weight sharing is difficult to reconcile with biologically local plasticity, because it would require distant synapses to maintain identical values without a known mechanism for transporting weights between them. Locally connected networks (LCNs) preserve spatially restricted receptive fields but assign independent filters to different locations, thereby removing the weight-sharing requirement. The resulting flexibility comes with an empirical challenge: biologically motivated learning methods have generally scaled less successfully in locally connected architectures than backpropagation in convolutional models on demanding visual tasks (Bartunov et al., 2018). Pogodin et al. showed that lateral interactions and a sleep-like Hebbian phase can coordinate initially unshared local filters and recover performance close to that of convolutional networks (Pogodin et al., 2021). The CorInfoMax experiments below pursue the same broad objective through a different normative mechanism, combining local receptive fields with correlative-information and score-broadcast updates without imposing exact convolutional weight tying.

G.3.1 The locally connected CorInfoMax experiment

The locally connected CorInfoMax implementation contains eight spatially locally connected stages, followed by a dense 512-unit hidden stage and a dense ten-class output stage: nine hidden stages and ten trainable transformations in total. Its architecture vector is

$$[3072,\ 16384,\ 2048,\ 512,\ 1024,\ 1024,\ 512,\ 768,\ 512,\ 512,\ 10]. \tag{32}$$

The detailed LC topology is:

Stage	Input shape	Output shape	Kernel	Stride	Padding
LC0	$3 \times 32 \times 32$	$64 \times 16 \times 16$	5	2	2
LC1	$64 \times 16 \times 16$	$32 \times 8 \times 8$	5	2	2
LC2	$32 \times 8 \times 8$	$32 \times 4 \times 4$	2	2	0
LC3	$32 \times 4 \times 4$	$64 \times 4 \times 4$	3	1	1
LC4	$64 \times 4 \times 4$	$64 \times 4 \times 4$	3	1	1
LC5	$64 \times 4 \times 4$	$128 \times 2 \times 2$	3	2	1
LC6	$128 \times 2 \times 2$	$192 \times 2 \times 2$	1	1	0
LC7	$192 \times 2 \times 2$	$128 \times 2 \times 2$	1	1	0

The resulting $128 \times 2 \times 2 = 512$ representation is processed by a dense $512 \to 512$ stage and a dense $512 \to 10$ classifier. LC6 and LC7 are unshared pointwise channel transformations at each spatial position; they increase depth without expanding the spatial support.

The lateral kernels are $(5, 5, 2, 3, 3, 3, 3, 3)$ for LC0–LC7, with same-channel lateral connectivity. Feedforward, feedback, and lateral updates use the LC index maps, so their outer products remain local. The run uses the raw, uncentered ten-dimensional CE score and a direct CE classifier update without momentum filtering. It performs three free neural-dynamics iterations per minibatch, uses batch size one, and disables the active feedback pathway with `include_backv=0`. Learned biases enter the hidden projection as $\mathbf{h} = \mathrm{clip}(\mathbf{u} + \mathbf{b}, 0, 50)$, while the plasticity gate is evaluated at $\mathbf{u}$ before adding the bias. Thus this gate is a local surrogate for the biased projection's derivative.

Figure 11 shows the train and test accuracy trajectories for the locally connected CorInfoMax-SBD model. Test accuracy settles around 72%, whereas training accuracy reaches 77.5%. The standard-deviation envelopes are widest during the early transition and become narrower later in training; the increasing train–test separation indicates a growing generalization gap after the initial rapid improvement.

The table lists reference coefficients, rather than effective parameter step sizes: they are combined with the feedforward step scale, cosine schedule, and activity-dependent controller in the supplied implementation. The classifier additionally scales its direct CE update by 0.0025. Both CorInfoMax experiments use the coupled schedule to warm up the step size, hold it through epoch 20, and then decay to half its reference scale.

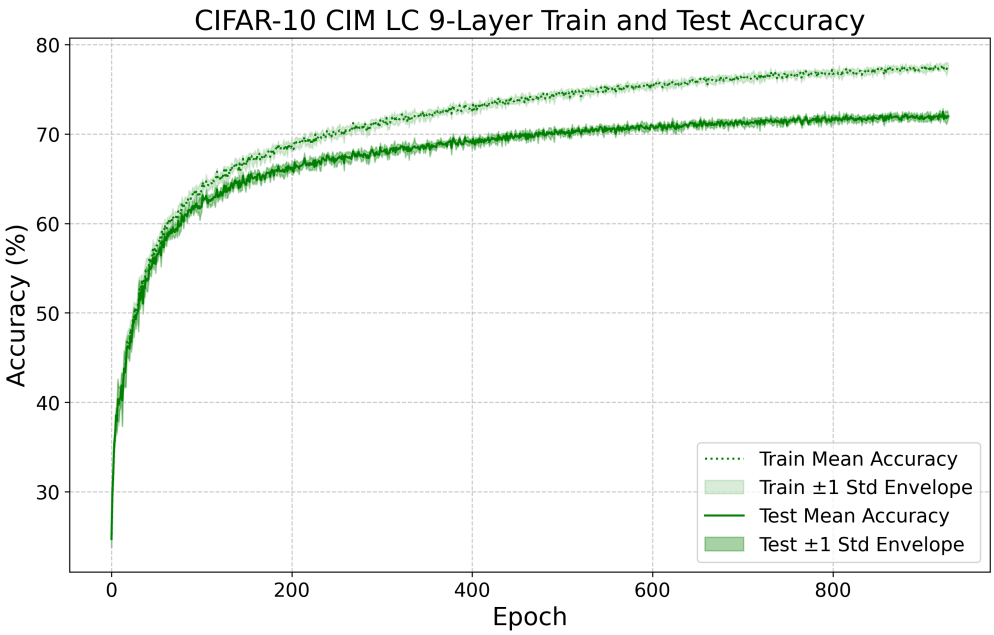


Figure 11: Training and test accuracy of the locally connected CorInfoMax-SBD model on CIFAR-10. Curves show the mean over five independent seed runs, and shaded regions indicate one standard deviation. The dotted and solid curves denote training and test accuracy, respectively.

Table 15: Hyperparameters for the locally connected CorInfoMax network.

Quantity	**Value**
Dataset and batch	CIFAR-10; $B = 1$
LC stages	8 unshared stages; dense $512 \to 512 \to 10$ head
Objective	Cross-entropy
Free neural iterations	$T = 3$
Lateral kernel sizes	$(5, 5, 2, 3, 3, 3, 3, 3)$; same-channel lateral maps
Lateral broadcast reference coefficient	0.32
Supervised update reference coefficients	$(50, 50, 50, 35, 30, 25, 20, 15, 50, 60000)$
Forgetting factors	$\lambda_{\mathrm{eb}} : 0.999999 \to 0.99999999$; $\lambda = 0.999999949999981$
Learning-rate schedule	Coupled cosine; warmup 1, decay $20 \to 800$ epochs, floor 0.50
Hidden-update momentum	Increases from 0.99 toward 0.999
Initialization	LC generic Xavier; local lateral identity initialization disabled; fan-aware initialization disabled
Augmentation	Enabled from epoch 0; constant padding 4 and horizontal flip
Activity controls	Power ceilings $(2, 12, 14, 4, 4, 4, 4, 4, 12)$; layer targets $(1, 1, 1, 1.25, 1.25, 1.5, 1.5, 1.5, 2, 0.1)$

H SBD IN A RESIDUAL NETWORK

We evaluate an SBD residual-network variant on a wide CIFAR ResNet-20. Hidden-layer learning uses the score-decorrelation construction of Appendices C.2 and C.3, with pooled activation features, contextual score expansion, and local update scaling. This appendix specifies the architecture, the activation transform $\mathbf{g}^{(k)}$ used in its score correlations, the local derivatives through residual addition and batch normalization (BN), and the expanded score and training settings used in this experiment. Hidden layers use only the local score-decorrelation signals described below; there is no auxiliary supervised hidden-layer loss.

H.1 Residual architecture and local updates

We use post-activation residual blocks (He et al., 2016) with three blocks per stage and widths $128, 256, 512$, respectively (Table 16). This is an eightfold widening of the CIFAR architecture, including its stem, rather than the standard wide-residual-network configuration (Zagoruyko and Komodakis, 2016). We retain u for preactivations. In the block equations below, $\mathbf{u}$ denotes the raw convolutional preactivation tensor; $\widehat{\mathbf{u}}$ and $\widetilde{\mathbf{u}}$ denote its normalized value and BN output, respectively (Eq. (38)). We write f for the elementwise ReLU activation, $f(a) = \max\{0, a\}$, with $f'(0) = 0$. For a block input $\mathbf{x}$,

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{x} * \mathbf{W}_1, & \widetilde{\mathbf{u}}_1 &= \mathrm{BN}_1(\mathbf{u}_1), & \mathbf{H}_1 &= f(\widetilde{\mathbf{u}}_1), \\ \mathbf{u}_2 &= \mathbf{H}_1 * \mathbf{W}_2, & \widetilde{\mathbf{u}}_2 &= \mathrm{BN}_2(\mathbf{u}_2), & \mathbf{H} &= f(\mathbf{s} + \widetilde{\mathbf{u}}_2). \end{aligned} \tag{33}$$

Both main convolutions have 3×3 kernels and padding one. The shortcut is $\mathbf{s} = \mathbf{x}$, except at stage transitions, where $\mathbf{u}_s = \mathbf{x} * \mathbf{W}_s$ and $\mathbf{s} = \widetilde{\mathbf{u}}_s = \mathrm{BN}_s(\mathbf{u}_s)$ define a learned 1×1 projection followed by BN. Both the first main convolution and the projection have stride two at stage transitions; all other strides are one. There is no activation between the second BN and addition.

Table 16: Wide CIFAR ResNet-20. Shapes exclude the minibatch dimension.

Part	Operation	Output shape
Input	Normalized RGB image	$3 \times 32 \times 32$
Stem	3×3 convolution; BN; f	$128 \times 32 \times 32$
Stage 1	Three residual blocks	$128 \times 32 \times 32$
Stage 2	Three blocks; first downsamples	$256 \times 16 \times 16$
Stage 3	Three blocks; first downsamples	$512 \times 8 \times 8$
Readout	Global average pooling; affine classifier	10 logits

Convolutions are bias-free; BN has learned scale and offset, and the classifier has a bias. There is no initial max-pooling layer. The network has $17{,}289{,}866$ trainable parameters. Its conventional depth count comprises the stem, 18 main-path convolutions, and classifier; the two projection convolutions give 21 trainable convolutions in total, indexed by k below. Convolutional weights use Kaiming-normal initialization with fan-in scaling for ReLU (He et al., 2015). Classifier weights have independent $\mathcal{N}(0, 1/512)$ entries, and its bias starts at zero. BN initialization is specified in Section H.2.

Score–activation correlations. Each convolution maintains its own running correlation $\widehat{\widetilde{\mathbf{R}}}^{(k)}$ between transformed activations $\mathbf{g}^{(k)}(\mathbf{H}^{(k)})$ and the expanded output score $\widetilde{\boldsymbol{\delta}}$, following Appendix D. The expanded score is defined in Section H.3. For the stem and each first block convolution, $\mathbf{H}^{(k)}$ is the convolution's own activation after BN and f. The second convolution and the projection instead use the merged block output $\mathbf{H}$ in Eq. (33), with a separate correlation matrix for each convolution.

For example i, write $\mathbf{H}_i^{(k)} \in \mathbb{R}^{P^{(k)} \times M^{(k)} \times N^{(k)}}$, using the channel and spatial dimensions of Appendix C.3. Here $\mathbf{g}^{(k)}$ combines spatial averages of the activations and their squares. Let A_4 average each channel into a 4×4 grid, and set

$$\mathbf{g}^{(k)}(\mathbf{H}) = \mathrm{vec}\left[A_4\mathbf{H};\ \frac{\xi}{\sqrt{2}} A_4(\mathbf{H} \odot \mathbf{H})\right], \qquad \xi = 0.5. \tag{34}$$

Each channel contributes sixteen averages and sixteen mean-square values, giving $32P^{(k)}$ coordinates per example. Squaring is performed before pooling. As in Appendix C.2, collect examples as columns of $\mathbf{G}^{(k)} \in \mathbb{R}^{32P^{(k)} \times B}$ and $\widetilde{\boldsymbol{\Delta}} \in \mathbb{R}^{110 \times B}$. At minibatch $m = 1, 2, \ldots$, update the correlation estimate and then project the expanded score:

$$\begin{aligned} \widehat{\widetilde{\mathbf{R}}}^{(k)}[m] &= \lambda \widehat{\widetilde{\mathbf{R}}}^{(k)}[m-1] + \frac{1-\lambda}{B} \mathbf{G}^{(k)}[m] \widetilde{\boldsymbol{\Delta}}[m]^\top, \\ \widetilde{\mathbf{Q}}^{(k)}[m] &= \widehat{\widetilde{\mathbf{R}}}^{(k)}[m] \mathbf{M}[m] \widetilde{\boldsymbol{\Delta}}[m]. \end{aligned} \tag{35}$$

The matrices $\widehat{\widetilde{\mathbf{R}}}^{(k)} \in \mathbb{R}^{32P^{(k)}\times 110}$ are independently initialized with $\mathcal{N}(0, 0.01^2)$ entries at $m = 0$. These are uncentered correlations, as in the preceding SBD derivation. Each projected score uses the updated full-minibatch correlation, including that example's own contribution. The shared positive diagonal matrix $\mathbf{M}$ rescales the score coordinates and is specified in Section H.4. We suppress the minibatch index below. Using the scaling factor c_k specified in Section H.4, the local updates use the following weighted and rescaled version of the expanded-score objective in Appendix D:

$$\widetilde{\mathcal{J}}^{(k)}_{\text{Score}} = \frac{c_k}{2(1-\lambda)M^{(k)}N^{(k)}} \operatorname{tr}\left(\widehat{\widetilde{\mathbf{R}}}^{(k)} \mathbf{M} \widehat{\widetilde{\mathbf{R}}}^{(k)\top}\right).$$

Reshape the two parts of column i of $\widetilde{\mathbf{Q}}^{(k)}$ into $P^{(k)} \times 4 \times 4$ grids, denoted $\widetilde{\mathbf{Q}}^{(k)}_{i,\text{lin}}$ and $\widetilde{\mathbf{Q}}^{(k)}_{i,\text{sq}}$. Differentiating $\mathbf{g}^{(k)}$ gives the direction at the full activation map:

$$\mathbf{q}^{(k)}_{H,i} = A_4^* \widetilde{\mathbf{Q}}^{(k)}_{i,\text{lin}} + \sqrt{2}\,\xi\, \mathbf{H}^{(k)}_i \odot A_4^* \widetilde{\mathbf{Q}}^{(k)}_{i,\text{sq}}. \tag{36}$$

Here A_4^* copies each grid value to the positions in its pooling cell and divides by the cell area. The second term includes the derivative of the squared activations. Thus $\mathbf{q}^{(k)}_H$, stacked over the minibatch, replaces $\mathbf{g}'^{(k)}(\mathbf{h}^{(k)}) \odot \widetilde{\mathbf{q}}^{(k)}$ from the elementwise derivation: it includes the derivative of $\mathbf{g}^{(k)}$, before the activation and BN derivatives.

We use the local-derivative notation of Appendix C.2 throughout: the subscript local means that only the dependence through the current convolution's activations is differentiated. The broadcast score, previous correlation estimate, convolution input, $\mathbf{M}$, and c_k are held fixed. The objective's scaling converts the correlation derivative's prefactor $(1-\lambda)/B$ to $c_k/[BM^{(k)}N^{(k)}]$. Thus the local gradients average over batch and spatial positions, with no channel-count divisor. This reduction is additional to the cell-area division in A_4^*.

Local derivatives through the residual block. For one channel and batch/spatial position n, the local activation derivatives are

$$\begin{aligned} \frac{\partial H_{1,n}}{\partial \widetilde{u}_{1,n}} &= f'(\widetilde{u}_{1,n}), \\ \frac{\partial H_n}{\partial \widetilde{u}_{2,n}} = \frac{\partial H_n}{\partial \widetilde{u}_{s,n}} &= f'(s_n + \widetilde{u}_{2,n}). \end{aligned} \tag{37}$$

Each convolution uses its own projected score. The second convolution holds $\mathbf{H}_1, \mathbf{s}$ fixed, and the projection holds $\mathbf{x}, \widetilde{\mathbf{u}}_2$ fixed. Addition has derivative one with respect to either branch, so both evaluate f' at the summed input. The first convolution instead receives its own signal at $\mathbf{H}_1$; the second convolution's signal does not propagate through it. An identity shortcut has no parameters to update. The stem uses $f'(\widetilde{u}_n)$, as does the first convolution.

H.2 Batch normalization

We use standard BN (Ioffe and Szegedy, 2015), with one learned scale γ and offset β per channel. For one fixed channel of convolution k, flatten the example and spatial indices into $n, j = 1, \ldots, BM^{(k)}N^{(k)}$. Thus $H^{(k)}_n$ and $q^{(k)}_{H,n}$ denote scalar entries of $\mathbf{H}^{(k)}$ and $\mathbf{q}^{(k)}_H$ in this channel. Suppressing the channel and layer indices on the BN quantities, the forward computation is

$$\begin{aligned} \mu &= \frac{1}{BM^{(k)}N^{(k)}} \sum_n u_n, \qquad v = \frac{1}{BM^{(k)}N^{(k)}} \sum_n (u_n - \mu)^2, \\ \sigma &= \sqrt{v + \epsilon_{\text{BN}}}, \qquad \widehat{u}_n = \frac{u_n - \mu}{\sigma}, \qquad \widetilde{u}_n = \gamma \widehat{u}_n + \beta. \end{aligned} \tag{38}$$

Here $\epsilon_{\text{BN}} = 10^{-5}$, and the training variance uses the $BM^{(k)}N^{(k)}$ denominator.

Scale and offset updates. The update formulas below use $f'(\widetilde{u}_n)$, as appropriate for the stem and first block convolution. For the second convolution and projection, replace this factor throughout by $f'(s_n + \widetilde{u}_{2,n})$, including inside the sums in the convolutional weight update, following Eq. (37). All

BN quantities remain those of the convolution being updated. For the minibatch mean cross-entropy loss $\mathcal{L}_{\mathrm{CE}}$, backpropagation gives

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial \gamma} = \sum_n \frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial H_n^{(k)}} f'(\widetilde{u}_n)\widehat{u}_n, \qquad \frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial \beta} = \sum_n \frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial H_n^{(k)}} f'(\widetilde{u}_n),$$

since $\partial \widetilde{u}_n/\partial \gamma = \widehat{u}_n$ and $\partial \widetilde{u}_n/\partial \beta = 1$. Computing $\partial \mathcal{L}_{\mathrm{CE}}/\partial H_n^{(k)}$ requires differentiation through downstream layers. SBD instead uses the local derivative of the score-decorrelation objective,

$$\left.\frac{\partial \widetilde{\mathcal{J}}_{\mathrm{Score}}^{(k)}}{\partial H_n^{(k)}}\right|_{\mathrm{local}} = \frac{c_k}{BM^{(k)}N^{(k)}} q_{H,n}^{(k)},$$

giving

$$\begin{aligned}\left.\frac{\partial \widetilde{\mathcal{J}}_{\mathrm{Score}}^{(k)}}{\partial \gamma}\right|_{\mathrm{local}} &= \frac{c_k}{BM^{(k)}N^{(k)}} \sum_n q_{H,n}^{(k)} f'(\widetilde{u}_n)\widehat{u}_n, \\ \left.\frac{\partial \widetilde{\mathcal{J}}_{\mathrm{Score}}^{(k)}}{\partial \beta}\right|_{\mathrm{local}} &= \frac{c_k}{BM^{(k)}N^{(k)}} \sum_n q_{H,n}^{(k)} f'(\widetilde{u}_n).\end{aligned} \tag{39}$$

These local score-decorrelation gradients are supplied to the optimizer for the scale and offset updates. The sums receive only the displayed $c_k/[BM^{(k)}N^{(k)}]$ factor; no further averaging is applied.

Convolutional weight updates. The convolutional weights determine u, before normalization, so their gradients require continuing the chain rule through $\widehat{u}$. Changing one u_j changes the channel mean and variance, and therefore can affect every $\widehat{u}_n$ in that channel. Because $\partial\mu/\partial u_j = 1/[BM^{(k)}N^{(k)}]$ and $\partial v/\partial u_j = 2(u_j - \mu)/[BM^{(k)}N^{(k)}]$,

$$\frac{\partial \widehat{u}_n}{\partial u_j} = \frac{1}{\sigma}\left[\mathbf{1}\{n=j\} - \frac{1}{BM^{(k)}N^{(k)}} - \frac{\widehat{u}_n\widehat{u}_j}{BM^{(k)}N^{(k)}}\right]. \tag{40}$$

Here $\mathbf{1}\{n=j\}$ is one when $n=j$ and zero otherwise. Using $\partial \widetilde{u}_n/\partial u_j = \gamma\, \partial \widehat{u}_n/\partial u_j$, the local derivative with respect to the raw convolution output is

$$\begin{aligned}\left.\frac{\partial \widetilde{\mathcal{J}}_{\mathrm{Score}}^{(k)}}{\partial u_j}\right|_{\mathrm{local}} = \frac{c_k\gamma}{BM^{(k)}N^{(k)}\sigma}\Bigg[& q_{H,j}^{(k)} f'(\widetilde{u}_j) \\ &- \frac{1}{BM^{(k)}N^{(k)}} \sum_{n=1}^{BM^{(k)}N^{(k)}} q_{H,n}^{(k)} f'(\widetilde{u}_n) \\ &- \frac{\widehat{u}_j}{BM^{(k)}N^{(k)}} \sum_{n=1}^{BM^{(k)}N^{(k)}} q_{H,n}^{(k)} f'(\widetilde{u}_n)\widehat{u}_n\Bigg].\end{aligned}$$

Both sums run over all batch and spatial positions in the same channel. The subtraction terms account for the dependence of the batch mean and variance on the convolution outputs; the expression is exact for finite ϵ_{BN}. The averages inside the brackets arise from the BN Jacobian and do not introduce another objective-reduction factor. For a filter coefficient in output channel p, with input channel h and kernel offsets a, b, the local gradient is then

$$\left.\frac{\partial \widetilde{\mathcal{J}}_{\mathrm{Score}}^{(k)}}{\partial W_{h,a,b}^{(k,p)}}\right|_{\mathrm{local}} = \sum_j \left.\frac{\partial \widetilde{\mathcal{J}}_{\mathrm{Score}}^{(k)}}{\partial u_j}\right|_{\mathrm{local}} \frac{\partial u_j}{\partial W_{h,a,b}^{(k,p)}}.$$

The last derivative is the corresponding cached input-patch entry, as in Appendix C.3. The inputs are $\mathbf{x}$, $\mathbf{H}_1$, and $\mathbf{x}$ for the first convolution, second convolution, and projection, respectively. No additional batch or spatial averaging is applied. Local differentiation stops at these cached inputs.

BN initializes $\gamma = 1, \beta = 0$, with running mean zero and running variance one. Once per training forward pass, the running statistics use new-observation weight 0.1: the mean receives μ, and the variance receives $BM^{(k)}N^{(k)}v/[BM^{(k)}N^{(k)} - 1]$. This unbiased correction is used only for the running variance; training normalization and its derivatives use v. Local differentiation uses the cached BN state from the same forward pass. Evaluation uses the fixed running statistics.

H.3 CONTEXTUAL SCORE EXPANSION

For example i, let $\mathbf{a}_i$ be the logits, $\mathbf{p}_i = \mathrm{softmax}(\mathbf{a}_i)$, and $\mathbf{y}_i$ the one-hot label vector. The raw ten-class CE score is $\boldsymbol{\delta}_i = \mathbf{p}_i - \mathbf{y}_i$, at unit temperature. We use the deterministic-modulator construction of Appendix D, with modulators obtained from the predicted probabilities, the input image, and the final hidden representation.

Let $\mathbf{x}_{4,i} \in \mathbb{R}^{48}$ be the flattened 4×4 average-pooled, normalized RGB input, and let $\mathbf{h}_{\mathrm{pool},i} \in \mathbb{R}^{512}$ be the final global-average-pooled representation immediately before the classifier. Normalize each vector by its own root-mean-square magnitude, concatenate them into $\mathbf{c}_i \in \mathbb{R}^{560}$:

$$\begin{aligned} \mathrm{rn}(\mathbf{z}) &= \frac{\mathbf{z}}{\sqrt{\|\mathbf{z}\|_2^2/\dim(\mathbf{z}) + 10^{-5}}}, \\ \mathbf{c}_i &= [\mathrm{rn}(\mathbf{x}_{4,i}); \mathrm{rn}(\mathbf{h}_{\mathrm{pool},i})]. \end{aligned} \tag{41}$$

The matrix $\mathbf{P}_{\mathrm{ctx}} \in \mathbb{R}^{560\times 8}$ is drawn once per run, with independent $\mathcal{N}(0, 1/560)$ entries, and is fixed thereafter. The normalization in Eq. (41) is per example and does not center the vectors.

Using the modulator notation of Appendix D, define eleven vectors $\boldsymbol{\phi}_{\ell,i} \in \mathbb{R}^{10}$:

$$\begin{aligned} &\boldsymbol{\phi}_{1,i} = \mathbf{1}_{10}, \\ &\boldsymbol{\phi}_{2,i} = \sqrt{10}(\mathbf{p}_i - \mathbf{1}_{10}/10), \qquad \boldsymbol{\phi}_{3,i} = \mathrm{roll}_5(\boldsymbol{\phi}_{2,i}), \\ &\boldsymbol{\phi}_{\ell,i} = \sqrt{2}\tanh\big([\mathbf{P}_{\mathrm{ctx}}^\top \mathbf{c}_i]_{\ell-3}\big)\,\mathbf{1}_{10}, \qquad \ell = 4,\dots,11. \end{aligned}$$

Here $\mathbf{1}_{10}$ is the all-ones vector, and roll_5 cyclically shifts a vector by five positions, as in Appendix D. Each of the last eight modulators repeats one bounded context scalar across all ten classes. These scalars vary across examples and as the hidden representation changes. The complete broadcast contains eleven ten-dimensional blocks, so the expanded dimension is $D_\varepsilon = 11 \times 10 = 110$:

$$\widetilde{\boldsymbol{\delta}}_i = \frac{1}{\sqrt{11}}\left[\boldsymbol{\phi}_{1,i} \odot \boldsymbol{\delta}_i; \dots; \boldsymbol{\phi}_{11,i} \odot \boldsymbol{\delta}_i\right] \in \mathbb{R}^{110}. \tag{42}$$

The first three blocks use the raw score, centered probabilities and their cyclic shift; the last eight provide context-weighted score–activation correlations. The common factor $1/\sqrt{11}$ controls the overall scale and preserves the orthogonality argument. All convolutions use the same expanded score, while the classifier uses the raw ten-dimensional score.

At fixed network parameters, the modulators depend only on the forward inputs, including the other minibatch inputs through BN. For independently sampled examples, the Bayes-optimal CE score satisfies $\mathbb{E}[\boldsymbol{\delta}_i^\star \mid X_1, \dots, X_B] = 0$, so the orthogonality argument of Appendix D applies with the full minibatch inputs conditioned on. In the local hidden-layer updates, the entire computed broadcast is held fixed: no derivatives are taken through $\mathbf{p}_i$, $\mathbf{h}_{\mathrm{pool},i}$, or $\boldsymbol{\phi}_{\ell,i}$. The derivatives of each convolution's own BN statistics are included as in Section H.2.

H.4 UPDATE SCALING

Before AdamW, $\mathbf{M}$ weights broadcast coordinates and c_k rescales each convolution's activation signal before local differentiation. AdamW separately maintains moments of the resulting parameter gradients.

Score-coordinate scaling. To downweight broadcast coordinates with larger mean-square magnitudes, track their second moments, initialized at $v_{\delta,j}[0] = 0$:

$$v_{\delta,j}[m] = 0.99 v_{\delta,j}[m-1] + \frac{0.01}{B}\sum_{i=1}^{B} \widetilde{\delta}_{j,i}[m]^2, \qquad j = 1,\dots,110.$$

Initially $\mathbf{M}[1] = \mathbf{I}_{110}$. Every 50 minibatches, after all convolutions have used $\mathbf{M}[m]$, form the bias-corrected values $\bar{v}_{\delta,j}[m] = v_{\delta,j}[m]/(1 - 0.99^m)$ and set

$$\begin{aligned} r[m] &= \max\left\{\frac{0.05}{110}\sum_{j=1}^{110}\bar{v}_{\delta,j}[m], 10^{-8}\right\}, \\ w_j[m] &= (\bar{v}_{\delta,j}[m] + r[m])^{-1}, \\ \mathbf{M}[m+1] &= \operatorname{diag}\left(\frac{\mathbf{w}[m]}{110^{-1}\sum_{j=1}^{110} w_j[m]}\right). \end{aligned} \tag{43}$$

Thus the inverse-second-moment weights have mean one. At other updates, $\mathbf{M}[m+1] = \mathbf{M}[m]$. Only these diagonal moments enter the weighting; the correlation matrices remain in the original score coordinates, with $\mathbf{M}$ applied when forming $\widetilde{\mathbf{Q}}^{(k)}$.

Convolution-wise scaling. For each convolution, normalize $\mathbf{q}_H^{(k)}$ by a running root-mean-square magnitude. Define its current mean square over all batch, channel, and spatial entries as $a_k[m] = \|\mathbf{q}_H^{(k)}[m]\|_F^2/[BP^{(k)}M^{(k)}N^{(k)}]$. Set $v_{q,k}[1] = a_k[1]$, then use

$$v_{q,k}[m] = 0.95v_{q,k}[m-1] + 0.05a_k[m] \quad (m \geq 2), \qquad c_k[m] = [\max\{v_{q,k}[m], 10^{-12}\}]^{-1/2}.$$

The current minibatch uses this updated c_k, before the activation gate, batch/spatial reduction, and BN derivatives. It normalizes the activation signal rather than the final parameter step. Both $\mathbf{M}$ and c_k are held fixed in local differentiation.

H.5 Training and accuracy curves

Standard PyTorch AdamW (Paszke et al., 2019; Loshchilov and Hutter, 2019) uses the local hidden-layer gradients and the classifier's minibatch CE gradient, all computed from the same forward pass before any parameter update. Table 17 gives the data and optimizer settings.

Table 17: Training settings for the residual-network experiment.

Setting	Value
Data	CIFAR-10: official 50,000 train / 10,000 test; no held-out validation subset
Training budget	400 epochs
Batches and precision	Train/evaluation batch sizes $64/256$; shuffled training batches; incomplete training batch dropped; FP32
Augmentation	Four-pixel reflection padding, random 32×32 crop, horizontal flip with probability 0.5
Normalization	RGB means $(0.4914, 0.4822, 0.4465)$; SDs $(0.2470, 0.2435, 0.2616)$
AdamW	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$
Weight decay	10^{-4} on convolution/classifier weights; none on BN parameters or classifier bias
Evaluation	All 10,000 test images; no augmentation or ensemble; running BN statistics

For fractional epoch $t \in [0, 400]$, the common learning-rate multiplier is

$$s(t) = \begin{cases} \max\{t/5, 10^{-4}\}, & 0 \leq t < 5, \\ 1, & 5 \leq t \leq 30, \\ 0.02 + 0.98[1 + \cos(\pi(t-30)/370)]/2, & 30 < t \leq 400. \end{cases} \tag{44}$$

Let $\tau_k = 0$ for the stem and $\tau_k = 1/3, 2/3, 1$ for stages 1–3, with projections assigned to their stage. The convolution learning rate and correlation retention factor are

$$\begin{aligned} \eta_k(t) &= 2 \times 10^{-4} s(t)(0.35 + 0.65\tau_k)\left[1 - 0.8(1-\tau_k)\operatorname{clip}\left(\frac{t-240}{160}, 0, 1\right)\right], \\ \lambda(t) &= 1 - 10^{-5}[0.1 + 0.9s(\max\{t, 5\})]. \end{aligned} \tag{45}$$

BN uses $0.1\eta_k(t)$, while the classifier uses $2 \times 10^{-3}s(t)$ without the depth factors. The depth factors lower earlier-layer rates, with a further reduction after epoch 240; the shared retention $\lambda(t)$ is applied per minibatch.

At the fixed epoch-400 checkpoint, the wide CIFAR ResNet-20 SBD variant reaches $83.39 \pm 0.90\%$ training accuracy and $75.90 \pm 0.60\%$ test accuracy (mean $\pm$ sample standard deviation across five independent runs). Figure 12 shows the corresponding trajectories.

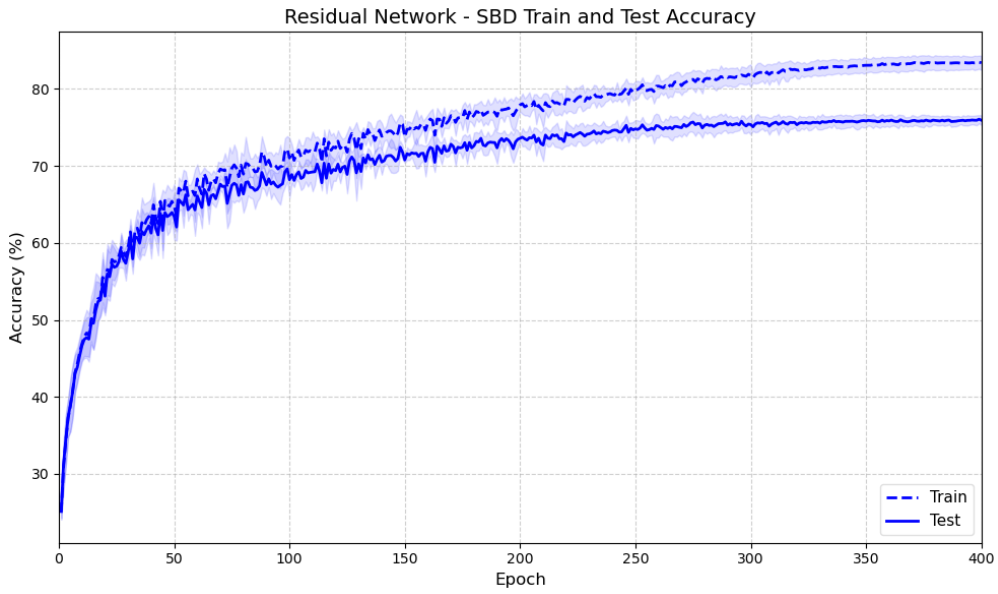


Figure 12: Training (dashed) and test (solid) accuracy of the wide CIFAR ResNet-20 SBD variant across 400 epochs. Curves show the mean across five runs; shaded regions show one standard deviation.