# Fast Gauss-Newton for Multiclass Cross-Entropy

**Mikalai Korbit** **Mario Zanon**
IMT School for Advanced Studies Lucca
Lucca, Italy
{mikalai.korbit,mario.zanon}@imtlucca.it

## Abstract

In multiclass softmax cross-entropy, the full generalized Gauss-Newton (GGN) curvature couples all output logits through the softmax covariance, making curvature-vector products harder to scale as the number of classes grows. We show that the standard multiclass GGN can be decomposed exactly into a true-vs-rest term and a positive semidefinite within-competitor covariance term. Fast Gauss-Newton (FGN) retains the first term and drops the second, yielding a positive semidefinite under-approximation of the multiclass GGN that is exact for binary classification. The derivation uses an exact true-vs-rest scalar-margin representation of softmax cross-entropy: the loss and gradient are unchanged, and the approximation enters only at the curvature level. Exploiting the FGN curvature structure, the damped update can be written as an equivalent whitened row-space system with one row per mini-batch example. We solve this system matrix-free by conjugate gradient using Jacobian-vector and vector-Jacobian products of the scalar margin map. Targeted mechanism experiments show how the dropped covariance grows with competitor dispersion and how damping can reduce its effect on the resulting steps. In frozen-feature head optimization and end-to-end CIFAR-100 training, FGN achieves similar test accuracy to a full GGN baseline with faster updates.

## 1 Introduction

The behavior and computational tractability of second-order methods are fundamentally governed by the structure of their curvature models. For neural network losses, the generalized Gauss-Newton (GGN) matrix provides a canonical positive semidefinite curvature approximation. In multiclass softmax cross-entropy, however, it inherits the full $C$-dimensional output geometry. For logits $\boldsymbol{z}(\boldsymbol{w}) \in \mathbb{R}^C$, let $\boldsymbol{J}_z \in \mathbb{R}^{C\times d}$ be the logit Jacobian with respect to the parameters and let $\boldsymbol{p} \in \mathbb{R}^C$ be the softmax probability vector. The corresponding GGN matrix is

$$\boldsymbol{J}_z^\top \big(\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top\big)\boldsymbol{J}_z,$$

so even matrix-free solvers still have to apply the $C$-dimensional softmax covariance in their curvature-vector products unless the output-space curvature model itself is changed [Schraudolph, 2002, Martens et al., 2010, Bottou et al., 2018]. We address this bottleneck by changing the curvature construction while leaving the softmax loss and gradient unchanged.

The central identity behind FGN is an exact decomposition of the multiclass softmax GGN:

$$\boldsymbol{H}^{\mathrm{GGN}} = \boldsymbol{H}^{\mathrm{FGN}} + \boldsymbol{R}^{\mathrm{comp}}, \qquad \boldsymbol{R}^{\mathrm{comp}} \succeq 0.$$

Here $\boldsymbol{H}^{\mathrm{FGN}}$ is the retained true-vs-rest curvature term, and $\boldsymbol{R}^{\mathrm{comp}}$ is a positive semidefinite (PSD) covariance term over the competing logits. FGN keeps the first term and drops the second, yielding a PSD under-approximation of the multiclass GGN that is exact in the binary case.

The retained term comes from an exact scalar true-vs-rest representation of the softmax cross-entropy, defined formally in Section 2. This yields one Jacobian row of the scalar margin per example, making

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the damped FGN update reducible to a $b$-dimensional row-space solve. The resulting matrix-free CG solver retains the structure of Gauss-Newton methods while avoiding the class-coupled curvature products required by the full softmax GGN.

FGN is a structured, regime-dependent curvature approximation. The approximation is tighter when one clear hardest negative dominates the competitor distribution, so that the dropped covariance term is small. When probability mass is spread across several competitors, $\boldsymbol{R}^{\mathrm{comp}}$ can be larger. This affects approximation tightness, not the validity of FGN: the decomposition holds for every finite-logit example.

This positioning distinguishes FGN from nearby second-order methods. In solver structure, FGN is closest to Hessian-free optimization [Martens and Sutskever, 2011, Wiesler et al., 2013, Kiros, 2013], stochastic Gauss-Newton methods such as SGN [Gargiani et al., 2020], and row-space formulations based on matrix inversion identities [Ren and Goldfarb, 2019, Korbit et al., 2025]. It shares the matrix-free linear algebra, but not the curvature construction: those methods still operate on the full multiclass output geometry. Compared with diagonal or structured preconditioners such as AdaHessian [Yao et al., 2021], Sophia [Liu et al., 2023], K-FAC [Martens and Grosse, 2015], and Shampoo [Gupta et al., 2018], FGN is not another factorization of the same softmax GGN/Fisher. Instead, it exploits a loss-specific decomposition that changes the output-space curvature before the row-space solve. Unlike sampled-output approximations or multiclass-to-binary training reductions [Chen et al., 2016, Allwein et al., 2000, Rifkin and Klautau, 2004], FGN leaves the original softmax loss and exact gradient unchanged; the approximation begins only at the GGN curvature level. Appendix A discusses these connections in more detail.

Our contributions are as follows.

- We derive an exact decomposition of the standard softmax GGN into a true-vs-rest term and a PSD covariance term over competing logits. We define FGN by retaining the true-vs-rest term while preserving the exact softmax loss and gradient. The resulting curvature is a PSD under-approximation of the multiclass GGN, is exact in the binary case, and differs from the full GGN by an explicit weighted covariance of competitor Jacobian rows.
- We derive a whitened $b$-dimensional row-space system for the damped FGN step and a matrix-free conjugate gradient solver based on scalar-margin Jacobian-vector and vector-Jacobian products.
- We test what the approximation discards and what it gains: how the dropped covariance affects damped steps, and whether the resulting solver can reduce update cost while preserving test accuracy, from frozen-feature head optimization to end-to-end ResNet-20 training on CIFAR-100.

## 2 True-vs-Rest Margin Setup

We use a standard reformulation of multiclass softmax cross-entropy as a softplus of a true-vs-rest margin; see, e.g., Kobayashi [2019]. This section introduces the notation and identities used in the curvature decomposition in Section 3.

Consider one training example $(\boldsymbol{x}, \boldsymbol{y})$, parameters $\boldsymbol{w} \in \mathbb{R}^d$, and $C \geq 2$ classes. Let $\boldsymbol{y} \in \{0,1\}^C$ be one-hot with true class denoted by $\star$. For class-indexed quantities, the subscript $-\star$ denotes the competitor block obtained by removing the true-class entry. The symbol $\dagger$ denotes an aggregate competitor quantity, not an additional class: $z_\dagger$ is the log-sum-exp aggregate of the competitor logits and $p_\dagger$ is their total probability mass.

The model outputs logits $\boldsymbol{z}(\boldsymbol{w}) = f(\boldsymbol{x}; \boldsymbol{w}) \in \mathbb{R}^C$, with class components $z_c(\boldsymbol{w})$ for $c = 1, \ldots, C$; the induced probabilities are $\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{z})$. Gradients of scalar-valued functions are column vectors; their transposes are Jacobian rows. Unless a derivative with respect to logits is written explicitly as $\nabla_{\boldsymbol{z}}$, Jacobians are with respect to the parameters $\boldsymbol{w}$. The logit Jacobian is $\boldsymbol{J}_z(\boldsymbol{w}) \coloneqq \partial \boldsymbol{z}(\boldsymbol{w}) / \partial \boldsymbol{w} \in \mathbb{R}^{C \times d}$; its $c$-th row is $\nabla_{\boldsymbol{w}} z_c(\boldsymbol{w})^\top$. Throughout, logits are assumed finite, and derivative identities are understood at points where the required derivatives exist.

Define the aggregate competitor logit and aggregate competitor probability by

$$z_\dagger(\boldsymbol{w}) \coloneqq \log \sum_{j \neq \star} e^{z_j(\boldsymbol{w})}, \tag{1}$$

$$p_\dagger(\boldsymbol{w}) \coloneqq \sum_{j \neq \star} p_j(\boldsymbol{w}) = \frac{e^{z_\dagger(\boldsymbol{w})}}{e^{z_\star(\boldsymbol{w})} + e^{z_\dagger(\boldsymbol{w})}} = 1 - p_\star(\boldsymbol{w}), \tag{2}$$

and the true-vs-rest margin by

$$s(\boldsymbol{w}) \coloneqq z_\dagger(\boldsymbol{w}) - z_\star(\boldsymbol{w}). \tag{3}$$

With $\phi(s) \coloneqq \log(1 + e^s)$, the multiclass cross-entropy loss can be written exactly as a softplus of the true-vs-rest margin:

$$\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = -\sum_{c=1}^{C} y_c \log p_c(\boldsymbol{w}) = -\log p_\star(\boldsymbol{w}) = \log\bigl(1 + e^{s(\boldsymbol{w})}\bigr) = \phi(s(\boldsymbol{w})). \tag{4}$$

This is an exact representation of the loss: all competing logits still enter through the log-sum-exp aggregate $z_\dagger$.

To prepare the Gauss-Newton curvature and the gradient identity, we now differentiate the outer softplus link and the inner margin map. The derivatives of the softplus link are

$$\phi'(s) = \frac{e^s}{1 + e^s} = p_\dagger, \qquad \phi''(s) = \frac{e^s}{(1 + e^s)^2} = p_\star p_\dagger. \tag{5}$$

For finite logits, $p_\dagger > 0$. Define the conditional competitor distribution $\boldsymbol{\rho} \in \mathbb{R}^{C-1}$ by

$$\rho_j \coloneqq \frac{p_j}{p_\dagger}, \qquad j \neq \star. \tag{6}$$

Thus $\boldsymbol{\rho}$ is the softmax distribution restricted to the competitor classes and renormalized.

Order the logit coordinates as $(z_\star, \boldsymbol{z}_{-\star})$. Differentiating $z_\dagger$ and $s = z_\dagger - z_\star$ with respect to the logits gives

$$\nabla_{\boldsymbol{z}} z_\dagger = \begin{bmatrix} 0 \\ \boldsymbol{\rho} \end{bmatrix}, \qquad \nabla_{\boldsymbol{z}} s = \begin{bmatrix} -1 \\ \boldsymbol{\rho} \end{bmatrix}. \tag{7}$$

Equivalently,

$$p_\dagger \nabla_{\boldsymbol{z}} s = \boldsymbol{p} - \boldsymbol{y}, \tag{8}$$

where $\boldsymbol{p} - \boldsymbol{y}$ is written in the same reordered logit coordinates.

Define the margin Jacobian row

$$\boldsymbol{J}_s(\boldsymbol{w}) \coloneqq \nabla_{\boldsymbol{w}} s(\boldsymbol{w})^\top = (\nabla_{\boldsymbol{z}} s)^\top \boldsymbol{J}_z \in \mathbb{R}^{1 \times d}. \tag{9}$$

Using (7), this row has the explicit form

$$\boldsymbol{J}_s = -\nabla_{\boldsymbol{w}} z_\star^\top + \sum_{j \neq \star} \rho_j \, \nabla_{\boldsymbol{w}} z_j^\top. \tag{10}$$

Thus $\boldsymbol{J}_s$ is the $\boldsymbol{\rho}$-weighted mean competitor Jacobian row minus the true-class Jacobian row.

Combining the scalar chain rule with (8) gives

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \phi'(s) \nabla_{\boldsymbol{w}} s = p_\dagger \, \boldsymbol{J}_s^\top = \boldsymbol{J}_z^\top (\boldsymbol{p} - \boldsymbol{y}). \tag{11}$$

The final expression is exactly the standard softmax cross-entropy gradient. Hence the scalar-margin representation changes neither the softmax loss nor its gradient. Although the loss factors through the margin $s$, its geometry remains multiclass through $z_\dagger$ and $\boldsymbol{\rho}$. Section 3 uses this structure to define and decompose the FGN curvature; expanded algebra is provided in Appendix B.

## 3 FGN as a Structured Approximation to Multiclass GGN

### 3.1 Full softmax GGN and the retained margin term

The true-vs-rest reformulation preserves the softmax loss and gradient exactly. This does not mean it preserves the Gauss-Newton curvature: the latter depends on the choice of inner map. The standard multiclass GGN uses the full logit map; FGN uses the scalar margin. We now make their difference explicit.

For a single example, the classical multiclass GGN for softmax cross-entropy is

$$\boldsymbol{H}^{\mathrm{GGN}}(\boldsymbol{w}) \coloneqq \boldsymbol{J}_z^\top \big(\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top\big)\boldsymbol{J}_z, \tag{12}$$

where $\boldsymbol{J}_z \in \mathbb{R}^{C\times d}$ is the logit Jacobian from Section 2. The factor $\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top$ is the standard output-space curvature for multinomial log-likelihoods and coincides with the Fisher factor in logit space [Schraudolph, 2002, Bottou et al., 2018].

Applying the Gauss-Newton construction instead to the exact scalar composition $\ell(\boldsymbol{w}) = \phi(s(\boldsymbol{w}))$ from Section 2 yields

$$\boldsymbol{H}^{\mathrm{FGN}}(\boldsymbol{w}) \coloneqq \phi''\big(s(\boldsymbol{w})\big)\, \boldsymbol{J}_s(\boldsymbol{w})^\top \boldsymbol{J}_s(\boldsymbol{w}) = p_\star p_\dagger\, \boldsymbol{J}_s^\top \boldsymbol{J}_s. \tag{13}$$

Each example therefore contributes a PSD matrix of rank at most one, formed from the true-vs-rest margin Jacobian.

Two chain-rule decompositions of the exact Hessian make the distinction explicit:

$$\begin{aligned}
\nabla^2_{\boldsymbol{w}}\ell &= \underbrace{\boldsymbol{J}_z^\top \big(\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top\big)\boldsymbol{J}_z}_{\boldsymbol{H}^{\mathrm{GGN}}(\boldsymbol{w})} + \sum_{c=1}^{C} (p_c - y_c)\, \nabla^2_{\boldsymbol{w}} z_c, \\
\nabla^2_{\boldsymbol{w}}\ell &= \underbrace{p_\star p_\dagger\, \boldsymbol{J}_s^\top \boldsymbol{J}_s}_{\boldsymbol{H}^{\mathrm{FGN}}(\boldsymbol{w})} + p_\dagger\, \nabla^2_{\boldsymbol{w}} s.
\end{aligned}$$

Thus, relative to the exact Hessian, both matrices are Gauss-Newton curvatures obtained by dropping second-derivative terms, but they correspond to different inner maps: the full logit map $\boldsymbol{z}(\boldsymbol{w})$ and the scalar margin map $s(\boldsymbol{w})$. Their difference is therefore already present in output-space curvature, and the next subsection identifies it precisely.

### 3.2 Decomposition of the softmax GGN

Using the coordinate order $(z_\star, \boldsymbol{z}_{-\star})$ from Section 2, write

$$\boldsymbol{z} = \begin{bmatrix} z_\star \\ \boldsymbol{z}_{-\star} \end{bmatrix}, \qquad \boldsymbol{J}_z = \begin{bmatrix} \boldsymbol{J}_\star \\ \boldsymbol{J}_{-\star} \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} p_\star \\ \boldsymbol{p}_{-\star} \end{bmatrix} = \begin{bmatrix} p_\star \\ p_\dagger \boldsymbol{\rho} \end{bmatrix}.$$

Here $\boldsymbol{J}_\star \in \mathbb{R}^{1\times d}$ and $\boldsymbol{J}_{-\star} \in \mathbb{R}^{(C-1)\times d}$. By (7) and (9),

$$\nabla_{\boldsymbol{z}} s = \begin{bmatrix} -1 \\ \boldsymbol{\rho} \end{bmatrix}, \qquad \boldsymbol{J}_s = (\nabla_{\boldsymbol{z}} s)^\top \boldsymbol{J}_z = -\boldsymbol{J}_\star + \boldsymbol{\rho}^\top \boldsymbol{J}_{-\star}.$$

**Theorem 3.1** (Decomposition of the multiclass softmax GGN)**.** *For a single example with finite logits and $C \geq 2$, in the coordinate order $(z_\star, \boldsymbol{z}_{-\star})$, the softmax covariance admits the decomposition*

$$\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top = p_\star p_\dagger\, \nabla_{\boldsymbol{z}} s\, \nabla_{\boldsymbol{z}} s^\top + p_\dagger \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top \end{bmatrix}. \tag{14}$$

*Consequently,*

$$\boldsymbol{H}^{\mathrm{GGN}} = \boldsymbol{H}^{\mathrm{FGN}} + p_\dagger\, \boldsymbol{J}_{-\star}^\top \big(\mathrm{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top\big)\boldsymbol{J}_{-\star}. \tag{15}$$

*Proof.* Appendix C.2.

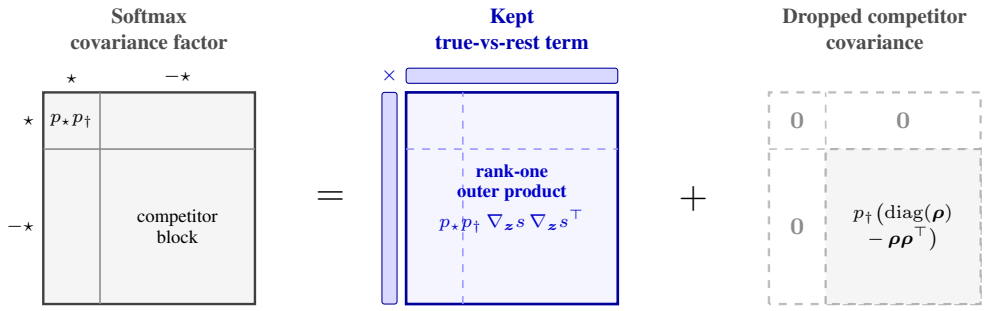



Figure 1: **Output-space decomposition underlying FGN.** The softmax covariance factor splits into a kept rank-one true-vs-rest term and a dropped within-competitor covariance term.

**Interpretation.** Theorem 3.1 identifies the approximation precisely. The kept term is the exact true-vs-rest covariance induced by the scalar margin $s = z_\dagger - z_\star$, while the residual is a covariance supported entirely on the competitor block. Figure 1 visualizes the output-space decomposition in Theorem 3.1; multiplication by $\boldsymbol{J}_z^\top$ and $\boldsymbol{J}_z$ yields the corresponding parameter-space decomposition. Thus FGN is not merely a cheaper implementation of the full softmax GGN: it is a different PSD curvature obtained by removing the within-competitor covariance term.

FGN, like full softmax GGN, retains all competitors. Since $\boldsymbol{J}_s = -\boldsymbol{J}_\star + \boldsymbol{\rho}^\top \boldsymbol{J}_{-\star}$, FGN uses all competitor logits through $\boldsymbol{\rho}$ and retains the exact $\boldsymbol{\rho}$-weighted mean competitor direction. This differs from a hard top-1 approximation, which would replace that mean by a single selected competitor. Near a tie, such a selection can change abruptly as the logits vary, whereas $\boldsymbol{\rho}$ and the corresponding weighted mean vary smoothly. Full GGN retains both the mean contribution and the within-competitor covariance; FGN drops only the latter.

### 3.3 Consequences: PSD under-approximation, exactness, and gap characterization

The decomposition has several immediate consequences.

**Proposition 3.2** (Consequences of the decomposition)**.** *For a single example with finite logits and $C \geq 2$, the following hold.*

1. $0 \preceq \boldsymbol{H}^{\mathrm{FGN}} \preceq \boldsymbol{H}^{\mathrm{GGN}}$.
2. *If $C = 2$, then $\boldsymbol{H}^{\mathrm{FGN}} = \boldsymbol{H}^{\mathrm{GGN}}$.*
3. *The dropped output-space residual in* (14) *vanishes if and only if $\boldsymbol{\rho}$ is one-hot.*
4. *If $\boldsymbol{J}_1, \ldots, \boldsymbol{J}_{C-1}$ are the rows of $\boldsymbol{J}_{-\star}$ and $\bar{\boldsymbol{J}}_\rho := \boldsymbol{\rho}^\top \boldsymbol{J}_{-\star}$, then*

$$\boldsymbol{H}^{\mathrm{GGN}} - \boldsymbol{H}^{\mathrm{FGN}} = p_\dagger \sum_{j=1}^{C-1} \rho_j \, (\boldsymbol{J}_j - \bar{\boldsymbol{J}}_\rho)^\top (\boldsymbol{J}_j - \bar{\boldsymbol{J}}_\rho). \tag{16}$$

*Proof.* Appendix C.3.

Item 1 shows that FGN is always a PSD under-approximation of the multiclass GGN. Item 2 is the binary special case. The third item characterizes exactness in output space. The dropped covariance vanishes exactly when the conditional competitor distribution $\boldsymbol{\rho}$ is one-hot; in that case the residual in (14) is already zero before projection, so $\boldsymbol{H}^{\mathrm{FGN}} = \boldsymbol{H}^{\mathrm{GGN}}$. For finite logits and $C > 2$, this condition is never attained because all competitor probabilities are positive, but it is approached as one competitor dominates. After projection to parameter space, equality can hold even when the competitor distribution is diffuse: the competitor Jacobian can annihilate a nonzero output-space residual, e.g., when all competitor Jacobian rows coincide. Finally, Item 4 identifies the dropped parameter-space term as a $\boldsymbol{\rho}$-weighted covariance of competitor Jacobian rows around their mean. The curvature gap therefore depends on both competitor probabilities and how differently their logits respond to parameter changes.

A useful per-example dispersion diagnostic is

$$\xi \coloneqq 1 - \|\boldsymbol{\rho}\|_2^2 = \mathrm{Tr}\big(\mathrm{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top\big). \tag{17}$$

It satisfies $0 \le \xi \le 1 - \frac{1}{C-1}$, with $\xi = 0$ if and only if $\boldsymbol{\rho}$ is one-hot, and the upper bound attained by the uniform competitor distribution.

For interpretation, define the effective number of competitors as

$$N_{\mathrm{eff}} \coloneqq \frac{1}{\|\boldsymbol{\rho}\|_2^2} = \frac{1}{1-\xi}.$$

If the conditional competitor mass is approximately uniform over a fixed subset of $k$ classes and negligible elsewhere, then $N_{\mathrm{eff}} \approx k$ and $\xi \approx 1 - 1/k$. Thus the total number of classes alone does not determine competitor dispersion.

The quantity $p_\dagger \xi$ is the exact trace of the dropped logit-space residual in (14). It separates two effects: $p_\dagger$ measures the total probability mass on the wrong classes, while $\xi$ indicates whether that mass is concentrated on one competitor or spread across many competitors. The dropped trace can therefore be small either because the model assigns high probability to the true class and $p_\dagger$ is small, or because one competitor dominates and $\xi$ is small. The parameter-space curvature gap $\boldsymbol{H}^{\mathrm{GGN}} - \boldsymbol{H}^{\mathrm{FGN}}$ further depends on the geometry of the competitor Jacobian rows through (16).

### 3.4 Batched form and row-space structure

The single-example identities above assemble cleanly at the mini-batch level. For a mini-batch $\mathcal{B} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^b$, define

$$\mathcal{L}(\boldsymbol{w}; \mathcal{B}) = \frac{1}{b} \sum_{i=1}^{b} \ell(\boldsymbol{w}; \boldsymbol{x}_i, \boldsymbol{y}_i), \tag{18}$$

and write $q_i(\boldsymbol{w}) \coloneqq p_\star^{(i)}(\boldsymbol{w}) p_\dagger^{(i)}(\boldsymbol{w})$. Collect the per-example quantities into

$$\boldsymbol{J}(\boldsymbol{w}) \coloneqq \begin{bmatrix} \nabla_{\boldsymbol{w}} s_1(\boldsymbol{w})^\top \\ \vdots \\ \nabla_{\boldsymbol{w}} s_b(\boldsymbol{w})^\top \end{bmatrix}, \qquad \boldsymbol{r}(\boldsymbol{w}) \coloneqq \begin{bmatrix} p_\dagger^{(1)}(\boldsymbol{w}) \\ \vdots \\ p_\dagger^{(b)}(\boldsymbol{w}) \end{bmatrix}, \qquad \boldsymbol{Q}(\boldsymbol{w}) \coloneqq \mathrm{diag}\big(q_1(\boldsymbol{w}), \dots, q_b(\boldsymbol{w})\big).$$

Then

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}; \mathcal{B}) = \frac{1}{b} \boldsymbol{J}(\boldsymbol{w})^\top \boldsymbol{r}(\boldsymbol{w}), \qquad \boldsymbol{H}^{\mathrm{FGN}}(\boldsymbol{w}; \mathcal{B}) = \frac{1}{b} \boldsymbol{J}(\boldsymbol{w})^\top \boldsymbol{Q}(\boldsymbol{w}) \boldsymbol{J}(\boldsymbol{w}). \tag{19}$$

For finite logits, $\boldsymbol{Q}(\boldsymbol{w})$ is diagonal and strictly positive, with entries in $(0, 1/4]$.

This batched factorization is the structural payoff of the decomposition. The mini-batch gradient remains exact, while the curvature acts entirely through the row space of $\boldsymbol{J}(\boldsymbol{w})$ with a diagonal positive middle factor $\boldsymbol{Q}(\boldsymbol{w})$. Unlike the softmax covariance $\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top$, which is singular when $C > 1$, the batch matrix $\boldsymbol{Q}(\boldsymbol{w})$ is diagonal and strictly positive for finite logits. This is the structure exploited by the whitened row-space solver in Section 4.

## 4 Matrix-Free Row-Space Solver for FGN

Starting from the batched factorization in (19), the damped FGN step can be reduced to a $b$-dimensional row-space solve. Whitening by $\boldsymbol{Q}(\boldsymbol{w})^{1/2}$ gives a symmetric positive definite (SPD) system that is directly compatible with matrix-free CG. The row-space reduction introduces no additional curvature approximation: the only approximation relative to full softmax GGN is the replacement of $\boldsymbol{H}^{\mathrm{GGN}}$ by $\boldsymbol{H}^{\mathrm{FGN}}$.

### 4.1 From the damped parameter-space system to the whitened row-space system

Fix the current parameters $\boldsymbol{w}$, mini-batch $\mathcal{B}$, and damping $\lambda > 0$, and let, for ease of notation,

$$\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{w}), \qquad \boldsymbol{r} = \boldsymbol{r}(\boldsymbol{w}), \qquad \boldsymbol{Q} = \boldsymbol{Q}(\boldsymbol{w}), \qquad \boldsymbol{g} = \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w};\mathcal{B}), \qquad \boldsymbol{H}^{\mathrm{FGN}} = \boldsymbol{H}^{\mathrm{FGN}}(\boldsymbol{w};\mathcal{B}).$$

Using (19), the damped FGN step is defined by

$$\big(\boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)\boldsymbol{d} = -\boldsymbol{g}. \tag{20}$$

Moreover,

$$\boldsymbol{g} \in \mathrm{range}(\boldsymbol{J}^\top), \qquad \mathrm{range}(\boldsymbol{H}^{\mathrm{FGN}}) \subseteq \mathrm{range}(\boldsymbol{J}^\top).$$

Therefore, for any $\lambda > 0$, the unique solution of (20) also lies in $\mathrm{range}(\boldsymbol{J}^\top)$. We parameterize the direction as

$$\boldsymbol{d} = -\boldsymbol{J}^\top \boldsymbol{\delta}, \qquad \boldsymbol{\delta} \in \mathbb{R}^b. \tag{21}$$

Substituting (21) into (20), and using $\boldsymbol{H}^{\mathrm{FGN}} = (1/b)\boldsymbol{J}^\top \boldsymbol{Q}\boldsymbol{J}$, gives

$$\big(\boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)\boldsymbol{d} = -\frac{1}{b}\,\boldsymbol{J}^\top\big(\boldsymbol{Q}\boldsymbol{J}\boldsymbol{J}^\top + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{\delta}. \tag{22}$$

Since $\boldsymbol{g} = (1/b)\boldsymbol{J}^\top \boldsymbol{r}$, it is sufficient to solve the unwhitened row-space system

$$\big(\boldsymbol{Q}\boldsymbol{J}\boldsymbol{J}^\top + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{\delta} = \boldsymbol{r}. \tag{23}$$

This system is generally non-symmetric. Since $\boldsymbol{Q}$ is diagonal with entries $q_i = p_\star^{(i)} p_\dagger^{(i)} \in (0, 1/4]$, $\boldsymbol{Q}^{\pm 1/2}$ are well defined and cheap to apply. Define the whitened quantities

$$\tilde{\boldsymbol{J}} \coloneqq \boldsymbol{Q}^{1/2}\boldsymbol{J}, \qquad \tilde{\boldsymbol{r}} \coloneqq \boldsymbol{Q}^{-1/2}\boldsymbol{r}, \qquad \boldsymbol{u} \coloneqq \boldsymbol{Q}^{-1/2}\boldsymbol{\delta}, \qquad \boldsymbol{K} \coloneqq \tilde{\boldsymbol{J}}\tilde{\boldsymbol{J}}^\top \in \mathbb{R}^{b\times b}. \tag{24}$$

Then the equivalent whitened system is

$$\big(\boldsymbol{K} + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{u} = \tilde{\boldsymbol{r}}. \tag{25}$$

Since $\boldsymbol{K} \succeq 0$ and $\lambda > 0$, the operator in (25) is symmetric positive definite and directly compatible with CG. The parameter-space direction is recovered by

$$\boldsymbol{d} = -\tilde{\boldsymbol{J}}^\top \boldsymbol{u} = -\boldsymbol{J}^\top \boldsymbol{Q}^{1/2}\boldsymbol{u}. \tag{26}$$

Indeed, if (25) holds, then

$$\big(\boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)(-\tilde{\boldsymbol{J}}^\top \boldsymbol{u}) = -\frac{1}{b}\tilde{\boldsymbol{J}}^\top\big(\boldsymbol{K} + b\lambda \boldsymbol{I}_b\big)\boldsymbol{u} = -\frac{1}{b}\tilde{\boldsymbol{J}}^\top \tilde{\boldsymbol{r}} = -\boldsymbol{g}.$$

Thus solving the whitened row-space system and backprojecting gives the exact damped FGN direction. Appendix D.1 gives the full parameter-space / row-space equivalence proof.

### 4.2 Matrix-free operator and cost

CG only requires products with the whitened row-space operator

$$\boldsymbol{B}\boldsymbol{v} \coloneqq \big(\boldsymbol{K} + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{v} = \tilde{\boldsymbol{J}}\big(\tilde{\boldsymbol{J}}^\top \boldsymbol{v}\big) + b\,\lambda \boldsymbol{v}, \qquad \boldsymbol{v} \in \mathbb{R}^b. \tag{27}$$

The matrix $\boldsymbol{B}$ is never materialized. One application can be implemented as

$$\boldsymbol{a} = \boldsymbol{Q}^{1/2}\boldsymbol{v}, \qquad \boldsymbol{\beta} = \boldsymbol{J}^\top \boldsymbol{a}, \qquad \boldsymbol{\gamma} = \boldsymbol{J}\boldsymbol{\beta}, \qquad \boldsymbol{B}\boldsymbol{v} = \boldsymbol{Q}^{1/2}\boldsymbol{\gamma} + b\,\lambda \boldsymbol{v}. \tag{28}$$

Thus each operator application uses one vector-Jacobian product (VJP) and one Jacobian-vector product (JVP) with the batched true-vs-rest margin map, plus $\mathcal{O}(b)$ vector operations. In standard row-space CG, the direction (26) is recovered after termination by one additional VJP.

In autodiff implementations, the batched margin map can be linearized once at the current parameters, and the resulting JVP/VJP closures can be reused across CG iterations. With zero initialization, at most $N_{\mathrm{CG}}$ CG iterations require at most $N_{\mathrm{CG}}$ applications of (27), each using one JVP and one VJP, plus $\mathcal{O}(N_{\mathrm{CG}} b)$ row-space vector operations and, in the standard implementation, one final VJP for backprojection.

A lifted implementation instead accumulates the parameter direction using the VJP already computed in each iteration, removing the final backprojection. It is equivalent to standard row-space CG at every iteration (in exact arithmetic), but requires storing and updating a $d$-dimensional vector.

FGN reduces the second-order solve to a $b$-dimensional row-space system and replaces class-coupled softmax covariance products with scalar-margin Jacobian products and diagonal reweighting. Full softmax GGN is also matrix-free, but each curvature product still propagates through all $C$ logits and applies the softmax covariance at a cost of $\mathcal{O}(C)$ per example. FGN still evaluates all $C$ logits and probabilities, so the complete training cost and memory remain class-dependent; its solve dimension and persistent row-space CG state, however, are independent of $C$. Algorithm 1 in Appendix E summarizes the core step, and Appendix D.2 compares its computational and storage requirements with full GGN, SGD, and Adam.

### 4.3 Solver interpretation and convergence properties

Let

$$\boldsymbol{A} \coloneqq \boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d, \qquad \boldsymbol{P} \coloneqq \boldsymbol{A}^{-1}.$$

With an exact inner solve, (20) gives $\boldsymbol{d} = -\boldsymbol{P}\boldsymbol{g}$, so the outer update $\boldsymbol{w}^+ = \boldsymbol{w} + \alpha \boldsymbol{d}$ has the form of a stochastic preconditioned gradient step with the FGN preconditioner. The row-space system is only the computational route used to apply $\boldsymbol{P}$ without forming $\boldsymbol{A}$.

With truncated CG, the same viewpoint holds with an inexact preconditioner application. If

$$\boldsymbol{e} \coloneqq \big(\boldsymbol{K} + b\lambda \boldsymbol{I}_b\big)\boldsymbol{u} - \tilde{\boldsymbol{r}}, \qquad \boldsymbol{d} \coloneqq -\tilde{\boldsymbol{J}}^\top \boldsymbol{u},$$

then the induced residual in the original damped parameter-space system obeys

$$\big(\boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)\boldsymbol{d} + \boldsymbol{g} = -\frac{1}{b}\tilde{\boldsymbol{J}}^\top \boldsymbol{e}, \qquad \big\|\big(\boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)\boldsymbol{d} + \boldsymbol{g}\big\| \le \frac{1}{b}\|\tilde{\boldsymbol{J}}\|\,\|\boldsymbol{e}\|. \tag{29}$$

An exact row-space solve ($\boldsymbol{e} = \boldsymbol{0}$) recovers the damped FGN direction, while (29) relates the row-space residual to the parameter-space residual.

Appendix D.3 provides a convergence analysis for a simplified preconditioned update with fixed damping and independent gradient and curvature mini-batches. For the empirical risk $\mathcal{R}$ defined there, assume that $\mathcal{R}$ is bounded below with a Lipschitz gradient, the stochastic gradients are unbiased with bounded second moments, the preconditioner has uniform positive lower and upper spectral bounds, and the positive step sizes satisfy $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$. Theorem D.6 then establishes almost-sure convergence of $\mathcal{R}(\boldsymbol{w}_t)$ to a finite limit and

$$\liminf_{t\to\infty} \mathbb{E}\big[\|\nabla \mathcal{R}(\boldsymbol{w}_t)\|^2\big] = 0.$$

Lemma D.7 extends these conclusions to inexact solves under its relative residual and summability conditions. The analysis follows standard stochastic preconditioning and inexact-solve arguments [Bottou et al., 2018, Li, 2017, Baey et al., 2023, Dembo et al., 1982, Bollapragada et al., 2019]. These guarantees do not cover the shared-batch Algorithm 1 or its practical extensions.

## 5 Experiments

We evaluate the FGN curvature approximation and its implementation in three settings. Section 5.1 examines update cost, the output-space decomposition in Theorem 3.1, and agreement between damped FGN and full GGN steps. Section 5.2 compares optimization of an affine head on frozen features, where full softmax GGN equals the Hessian of the loss. Section 5.3 evaluates end-to-end ResNet-20 training on CIFAR-100 with the same maximum CG iteration budget for SGN and FGN. We use SGN [Gargiani et al., 2020] as the full softmax GGN baseline, with damped steps computed using truncated CG. The FGN algorithm is implemented in Somax [Korbit and Zanon, 2026], a JAX library for second-order optimization. Scripts to reproduce the experimental results are available at `https://github.com/cor3bit/fgn`. Further experimental details are provided in Appendix F.

### 5.1 Mechanism checks

Panel (a) measures the cost of the implemented updates for an affine softmax head on fixed $2048$-dimensional Gaussian features, using batch size $512$ and $C = 2^k$ for $k = 1, \dots, 12$. SGN and FGN

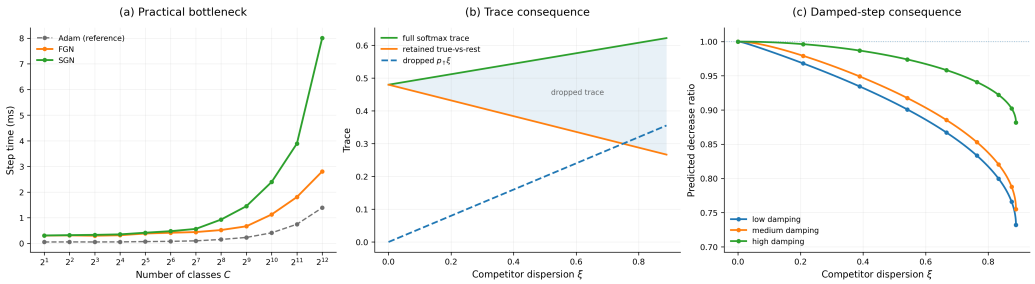
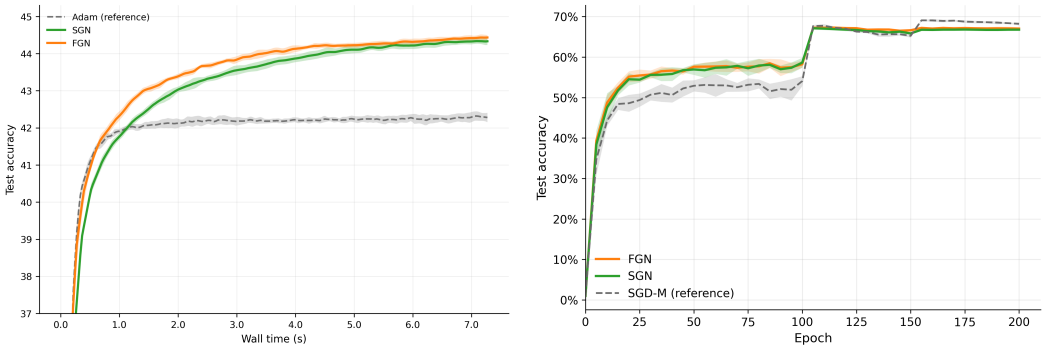



Figure 2: **Mechanism checks for FGN.** (a) Steady-state optimizer update time for a synthetic affine head, with at most 5 CG iterations for SGN and FGN. (b) Output-space traces at fixed $p_\star = 0.6$; the shaded gap is $p_\dagger\xi$. (c) Ratio of FGN to full GGN predicted decreases under the same damped quadratic model in a two-dimensional construction ($p_\star = 0.7$).




(a) Frozen-feature head optimization

(b) ResNet-20 on CIFAR-100

Figure 3: **FGN vs full GGN in multiclass classification.** (a) Cars196 frozen-feature head: test accuracy against head optimization wall time, excluding feature extraction, compilation, and evaluation. (b) ResNet-20 on CIFAR-100: test accuracy against training epoch. Curves show means over 10 seeds in (a) and 5 seeds in (b); shaded bands indicate $\pm 1$ standard deviation.

use fixed damping $\lambda = 1$ and at most five CG iterations; Adam [Kingma and Ba, 2014] provides a first-order cost reference. After compilation and warmup, FGN has lower update time than SGN, with a $2.85\times$ speedup at $C = 4096$. Both methods are matrix-free: SGN uses parameter-space CG with the full softmax curvature, whereas FGN uses row-space CG with a single true-vs-rest margin per example. Both methods evaluate all $C$ logits, and timings include complete optimizer updates.

Panel (b) visualizes the output-space trace decomposition in Theorem 3.1 for $C = 10$ and fixed $p_\star = 0.6$. The dropped trace $p_\dagger\xi$ increases linearly with competitor dispersion. The parameter-space gap additionally depends on the geometry of the competitor Jacobian rows through (16).

Panel (c) compares damped FGN and full GGN steps in a two-dimensional construction with affine logits ($C = 10$, $p_\star = 0.7$). Both steps are computed by direct solves and evaluated under the same full GGN damped quadratic model. The plotted ratio measures the predicted decrease achieved by the FGN step relative to the full GGN step. The ratio is close to one for concentrated competitor distributions and decreases as $\xi$ increases. Stronger damping reduces the influence of the dropped competitor covariance on the resulting steps, keeping the ratio closer to one.

### 5.2 Frozen-feature multiclass head optimization

To compare FGN with the exact Hessian, we train an affine softmax head on Cars196 [Krause et al., 2013], a fine-grained 196-class classification dataset. We use a frozen ImageNet-pretrained ResNet-50 [He et al., 2016] to extract fixed 2048-dimensional features $\boldsymbol{h}_i$, and train only the affine head $\boldsymbol{z}_i = \boldsymbol{W}^\top \boldsymbol{h}_i + \boldsymbol{b}$. Since the logits are affine in the optimized parameters, full softmax GGN equals the Hessian of the loss. SGN applies truncated CG in parameter space to the full GGN.

FGN uses a closed-form row-space Gram for the retained true-vs-rest curvature. For both methods, damping is fixed at $\lambda = 1$ and CG is capped at 5 iterations.

Figure 3(a) compares test accuracy against head optimization time, excluding feature extraction, compilation, and evaluation. Hyperparameters are selected on a validation split and then fixed for final test evaluation. At the end of the shared 7.27 s horizon, Adam, SGN, and FGN reach $42.28 \pm 0.13\%$, $44.33 \pm 0.10\%$, and $44.43 \pm 0.07\%$ test accuracy, respectively. FGN reaches $43\%$ and $44\%$ accuracy in 1.41 and 3.45 s, compared with 1.97 and 4.44 s for SGN. FGN achieves comparable test accuracy to SGN while reaching the same accuracy thresholds sooner. This occurs even though the dropped term accounts for $40.85\%$ of the full parameter-space GGN trace at the final FGN iterate (Appendix F.2).

### 5.3 End-to-end training

We train a CIFAR-style ResNet-20 [He et al., 2016] end-to-end on CIFAR-100 [Krizhevsky et al., 2009] to test whether the FGN under-approximation preserves optimization performance beyond the affine-head setting. We compare FGN with SGN, which uses the full softmax GGN, and include SGD with momentum (SGD-M) as a first-order reference. Unlike the frozen-feature head, the logits are now nonlinear in the optimized parameters.

SGD-M follows a standard CIFAR ResNet training recipe with 200 epochs, batch size 128, and an initial learning rate of 0.1, reduced by 10 at epochs 100 and 150. SGN and FGN follow the same decay schedule, starting from learning rate 0.3. Both curvature methods use fixed damping $\lambda = 1$, at most 5 CG iterations, and tolerance $10^{-5}$, with limited hyperparameter tuning.

Figure 3(b) shows similar test accuracy trajectories for FGN and SGN. After 200 epochs, FGN reaches $67.05 \pm 0.35\%$ test accuracy, compared with $66.78 \pm 0.23\%$ for SGN. The corresponding training step times are 76.15 and 95.50 ms, giving FGN a $1.25\times$ speedup. SGD-M is faster and achieves higher final test accuracy ($68.23 \pm 0.34\%$). SGN reaches lower training cross-entropy, while FGN has lower test cross-entropy (Appendix F.3, Table 4).

Thus, with the same CG iteration cap, FGN reaches similar test accuracy to SGN at lower update cost during nonlinear end-to-end training, despite discarding the within-competitor covariance.

## 6 Conclusion

We introduced Fast Gauss-Newton (FGN), a structured approximation to multiclass softmax GGN. FGN is based on an exact decomposition of GGN into a retained true-vs-rest term and a PSD within-competitor covariance residual. It preserves the original softmax loss and gradient, yields a batch row-space system solvable by matrix-free conjugate gradient, and recovers the full softmax GGN curvature in the binary case.

The approximation is regime-dependent: it is tighter when competitor mass is concentrated and can deviate more from full GGN when probability mass is spread across several competitors. Damping does not change this curvature gap, but can reduce its effect on the resulting step.

In both frozen-feature head optimization and end-to-end ResNet-20 training on CIFAR-100, FGN achieves test accuracy similar to the full GGN baseline with faster updates. A natural next step is to extend the true-vs-rest model with low-rank approximations of the within-competitor covariance.

## Acknowledgments and Disclosure of Funding

This work was funded by the Italian Ministero dell'Università e della Ricerca, project PRIN "Too much Information! Control-Oriented, Dynamic Information Selection" (Grant: 2022NLTTBN).

The authors have no competing interests to declare that are relevant to the content of this paper.

## References

Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, 1(Dec):113–141, 2000.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Charlotte Baey, Maud Delattre, Estelle Kuhn, Jean-Benoist Leger, and Sarah Lemler. Efficient preconditioned stochastic gradient descent for estimation in latent variable models. In *International Conference on Machine Learning*, pages 1430–1453. PMLR, 2023.

Yoshua Bengio and Jean-Sébastien Senécal. Quick training of probabilistic neural nets by importance sampling. In *International Workshop on Artificial Intelligence and Statistics*, pages 17–24. PMLR, 2003.

Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled Newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 2019.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In *International Conference on Machine Learning*, pages 557–565. PMLR, 2017.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Wenlin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1975–1985, 2016.

Shabnam Daghaghi, Tharun Medini, Nicholas Meisburger, Beidi Chen, Mengnan Zhao, and Anshumali Shrivastava. A tale of two efficient and informative negative sampling distributions. In *International conference on machine learning*, pages 2319–2329. PMLR, 2021.

Ron S Dembo, Stanley C Eisenstat, and Trond Steihaug. Inexact newton methods. *SIAM Journal on Numerical analysis*, 19(2):400–408, 1982.

Matilde Gargiani, Andrea Zanelli, Moritz Diehl, and Frank Hutter. On the promise of the stochastic generalized Gauss-Newton method for training DNNs. *arXiv preprint arXiv:2006.02409*, 2020.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

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*, pages 770–778, 2016.

Magnus R Hestenes, Eduard Stiefel, et al. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ryan Kiros. Training neural networks with stochastic Hessian-free optimization. *arXiv preprint arXiv:1301.3641*, 2013.

Takumi Kobayashi. Large margin in softmax cross-entropy loss. In *BMVC*, volume 2, page 3, 2019.

Mikalai Korbit and Mario Zanon. Second-order, first-class: A composable stack for curvature-aware training. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 90–105. Springer, 2026.

Mikalai Korbit, Adeyemi D Adeoye, Alberto Bemporad, and Mario Zanon. Exact gauss-newton optimization for training deep neural networks. *Neurocomputing*, page 131738, 2025.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.

Zhenyu Liao and Michael W Mahoney. Hessian eigenspectra of more realistic nonlinear models. *Advances in Neural Information Processing Systems*, 34:20104–20117, 2021.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*, 2023.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

James Martens and Ilya Sutskever. Learning recurrent neural networks with Hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1033–1040, 2011.

James Martens et al. Deep learning via Hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN 9780387400655.

Vardan Papyan. The full spectrum of deep net Hessians at scale: Dynamics with sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Yi Ren and Donald Goldfarb. Efficient subsampled gauss-newton and natural gradient methods for training neural networks. *arXiv preprint arXiv:1906.02353*, 2019.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing methods in statistics*, pages 233–257. Elsevier, 1971.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the Hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9481–9488, 2021.

Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Simon Wiesler, Jinyu Li, and Jian Xue. Investigations on Hessian-free optimization for cross-entropy training of deep neural networks. In *Interspeech*, pages 3317–3321, 2013.

Zeke Xie, Qian-Yuan Tang, Yunfeng Cai, Mingming Sun, and Ping Li. On the power-law hessian spectrums in deep learning, 2022. URL `https://arxiv.org/abs/2201.13011`.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10665–10673, 2021.

# A Related Work

This appendix reviews prior work around the paper's central technical choice: FGN modifies the *output-space* curvature model for softmax cross-entropy (via an exact true-vs-rest margin reformulation and a decomposition of the softmax GGN), and only then builds a row-space solver around the resulting batched form $\boldsymbol{H}^{\mathrm{FGN}}(\boldsymbol{w}; \mathcal{B}) = \frac{1}{b}\boldsymbol{J}^\top \boldsymbol{Q}\boldsymbol{J}$.

**Gauss-Newton, generalized Gauss-Newton, and Fisher curvature.** The Gauss-Newton and generalized Gauss-Newton matrices are standard PSD curvature approximations to the Hessian, obtained by linearizing the model and retaining only the curvature of the loss in *output space* [Schraudolph, 2002, Bottou et al., 2018]. For log-likelihood losses in exponential-family models, this output-space curvature is closely related to the Fisher information matrix, which also underlies natural gradient methods in information geometry [Amari, 1998, Martens, 2020]. This Fisher/GGN perspective is central to scalable second-order optimizers in deep learning. It motivates replacing the indefinite Hessian with a PSD curvature approximation and explains the common use of damping or trust-region style regularization for robustness in practice (e.g., via $(\boldsymbol{H} + \lambda \boldsymbol{I})\boldsymbol{d} = -\boldsymbol{g}$) [Martens, 2020, Nocedal and Wright, 2006]. Whereas much prior work retains the standard softmax output-space curvature and focuses on approximating or inverting the resulting parameter-space matrix, FGN modifies the output-space curvature factor itself while keeping the training loss and gradient exact.

**Softmax cross-entropy curvature and the output-space bottleneck.** For multiclass softmax cross-entropy, the canonical logit-space curvature factor is the $C \times C$ softmax covariance $\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top$, leading to a per-example GGN term $\boldsymbol{J}_z^\top(\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top)\boldsymbol{J}_z$ [Schraudolph, 2002, Martens, 2020]. Matrix-free GGN products avoid forming the parameter-space matrix, but still apply the softmax covariance to $C$-dimensional vectors. The core contribution of this paper is an *output-space* simplification that is specific to softmax cross-entropy: by reformulating the loss exactly as a scalar softplus of a true-vs-rest margin and then decomposing the softmax GGN, we isolate a true-vs-rest margin term and a PSD covariance term over competing logits. FGN retains the former and discards the latter. This differs from diagonal or structured approximations that keep the original softmax curvature model but compress the resulting parameter-space curvature.

**Output approximations and multiclass reductions.** Large-output models are often scaled by changing the training computation itself, for example through sampled or approximate output losses such as importance sampling, sampled softmax, target sampling, or negative sampling [Bengio and Senécal, 2003, Chen et al., 2016, Daghaghi et al., 2021]. Another approach reduces multiclass classification to collections of binary problems, including one-vs-rest schemes [Allwein et al., 2000, Rifkin and Klautau, 2004]. FGN is different from both families: it keeps the original softmax cross-entropy loss and exact gradient, and approximates only the GGN curvature factor.

**Second-order methods in deep learning.** A large literature studies second-order training methods whose main computational challenge is building and inverting curvature approximations. Hessian-free optimization and related truncated-Newton methods solve linear systems using conjugate gradients and matrix-free curvature-vector products computed by automatic differentiation [Martens et al., 2010, Martens and Sutskever, 2011, Pearlmutter, 1994, Hestenes et al., 1952]. These methods typically use the exact Hessian or GN/GGN curvature and introduce damping to stabilize the solve. Structured factorization approaches aim to retain richer geometry at lower cost. K-FAC approximates the Fisher/GGN by layerwise Kronecker factors [Martens and Grosse, 2015], while Shampoo exploits the tensor structure of parameter blocks and maintains separate preconditioners along each axis [Gupta et al., 2018]. Recent diagonal curvature methods such as AdaHessian and Sophia estimate (variants of) diagonal Hessian information at low overhead [Yao et al., 2021, Liu et al., 2023]. FGN is complementary to these methods: rather than introducing another layerwise factorization of the same softmax curvature, it modifies the *multiclass output-space* curvature model itself, yielding a different batched Jacobian structure before the linear solve.

**Stochastic Gauss-Newton and observation/row-space solvers.** Practical deep learning GN methods build tractable mini-batch GN/GGN approximations [Botev et al., 2017]. Given such approximations, stochastic GN methods typically solve the resulting damped linear systems only approximately, often by CG or related stochastic linear algebra procedures [Gargiani et al., 2020]. A separate

line of work exploits low-rank structure and observation-space identities to obtain solves that scale with batch size rather than parameter dimension [Ren and Goldfarb, 2019, Korbit et al., 2025]. These methods demonstrate that row-space solvers can be effective, but they typically retain the full multiclass output geometry in the curvature model. FGN targets the orthogonal bottleneck: the true-vs-rest decomposition gives one scalar-margin row per example, so the FGN row-space system is $b$-dimensional rather than tied to the full $bC$ logit block of the multiclass curvature.

**Empirical and theoretical structure of Hessian spectra.** Empirical studies of deep network Hessians report a characteristic spiked-plus-bulk structure, with a small number of outlier eigenvalues separated from a near-zero bulk [Papyan, 2018, Sagun et al., 2017]. Related work further studies layerwise and power-law spectral structure [Sankar et al., 2021, Xie et al., 2022, Liao and Mahoney, 2021]. These results motivate structured curvature models broadly, but they are not assumptions behind FGN. The FGN decomposition is instead loss-specific and algebraic: it separates a true-vs-rest margin direction from within-competitor covariance in logit space.

# B Derivations for the True-vs-Rest Margin Representation

This appendix verifies the identities used in Section 2, namely (4), (5), and (7).

## B.1 Exact scalar-margin representation

Starting from the single-example cross-entropy loss,

$$\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = -\log p_\star(\boldsymbol{w}) = -\log \frac{e^{z_\star}}{\sum_{j=1}^{C} e^{z_j}}, \tag{30}$$

split the denominator into the true-class term and the competing terms:

$$\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = -\log \frac{e^{z_\star}}{e^{z_\star} + \sum_{j \neq \star} e^{z_j}} = -z_\star + \log\Big(e^{z_\star} + \sum_{j \neq \star} e^{z_j}\Big). \tag{31}$$

Using $e^{z_\dagger} = \sum_{j \neq \star} e^{z_j}$, which follows from the definition of $z_\dagger$, we obtain

$$\begin{aligned} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) &= -z_\star + \log\big(e^{z_\star} + e^{z_\dagger}\big) = -z_\star + \log\Big(e^{z_\star}\big(1 + e^{z_\dagger - z_\star}\big)\Big) \\ &= -z_\star + z_\star + \log\big(1 + e^{z_\dagger - z_\star}\big) = \log\big(1 + e^{z_\dagger - z_\star}\big). \end{aligned} \tag{32}$$

Hence, with $s(\boldsymbol{w}) = z_\dagger(\boldsymbol{w}) - z_\star(\boldsymbol{w})$ and $\phi(s) = \log(1 + e^s)$, the loss has the exact scalar form

$$\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \phi\big(s(\boldsymbol{w})\big). \tag{33}$$

## B.2 Derivatives of the scalar link

Let

$$\phi(s) = \log(1 + e^s). \tag{34}$$

Then

$$\phi'(s) = \frac{e^s}{1 + e^s}, \qquad \phi''(s) = \frac{e^s}{(1 + e^s)^2}. \tag{35}$$

Substituting $s = z_\dagger - z_\star$ yields

$$\phi'(s) = \frac{e^{z_\dagger - z_\star}}{1 + e^{z_\dagger - z_\star}} = \frac{e^{z_\dagger}}{e^{z_\star} + e^{z_\dagger}} = p_\dagger, \tag{36}$$

$$\phi''(s) = \frac{e^{z_\dagger - z_\star}}{\big(1 + e^{z_\dagger - z_\star}\big)^2} = \frac{e^{z_\star} e^{z_\dagger}}{\big(e^{z_\star} + e^{z_\dagger}\big)^2} = p_\star p_\dagger. \tag{37}$$

### B.3 Logit-space derivatives of $z_\dagger$ and $s$

For convenience, order the logit coordinates as $(z_\star, \boldsymbol{z}_{-\star})$, with $\boldsymbol{\rho}$ as defined in (6). Since $z_\dagger = \log \sum_{j \neq \star} e^{z_j}$ does not depend on $z_\star$, we have

$$\frac{\partial z_\dagger}{\partial z_\star} = 0. \tag{38}$$

For $k \neq \star$,

$$\frac{\partial z_\dagger}{\partial z_k} = \frac{e^{z_k}}{\sum_{j \neq \star} e^{z_j}} = \frac{p_k}{p_\dagger} = \rho_k. \tag{39}$$

Therefore

$$\nabla_{\boldsymbol{z}} z_\dagger = \begin{bmatrix} 0 \\ \boldsymbol{\rho} \end{bmatrix}, \qquad \nabla_{\boldsymbol{z}} s = \nabla_{\boldsymbol{z}}(z_\dagger - z_\star) = \begin{bmatrix} -1 \\ \boldsymbol{\rho} \end{bmatrix}. \tag{40}$$

By the chain rule,

$$\nabla_{\boldsymbol{w}} z_\dagger = \sum_{j \neq \star} \rho_j \, \nabla_{\boldsymbol{w}} z_j, \qquad \nabla_{\boldsymbol{w}} s = -\nabla_{\boldsymbol{w}} z_\star + \sum_{j \neq \star} \rho_j \, \nabla_{\boldsymbol{w}} z_j. \tag{41}$$

Equivalently, the margin Jacobian row is

$$\boldsymbol{J}_s = \nabla_{\boldsymbol{w}} s^\top = -\nabla_{\boldsymbol{w}} z_\star^\top + \sum_{j \neq \star} \rho_j \, \nabla_{\boldsymbol{w}} z_j^\top. \tag{42}$$

### B.4 Equivalence of softmax and margin gradients

We show that the standard softmax cross-entropy gradient

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{J}_z(\boldsymbol{w})^\top \big(\boldsymbol{p}(\boldsymbol{w}) - \boldsymbol{y}\big) \tag{43}$$

coincides exactly with the margin form

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = p_\dagger(\boldsymbol{w}) \, \nabla_{\boldsymbol{w}} s(\boldsymbol{w}). \tag{44}$$

Using the reordered coordinates from Appendix B.3, the target and probability vectors have the block forms

$$\boldsymbol{y} = \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} p_\star \\ p_\dagger \boldsymbol{\rho} \end{bmatrix}. \tag{45}$$

Therefore

$$\boldsymbol{p} - \boldsymbol{y} = \begin{bmatrix} p_\star - 1 \\ p_\dagger \boldsymbol{\rho} \end{bmatrix} = p_\dagger \begin{bmatrix} -1 \\ \boldsymbol{\rho} \end{bmatrix} = p_\dagger \nabla_{\boldsymbol{z}} s. \tag{46}$$

Applying the chain rule gives

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \phi'(s) \, \nabla_{\boldsymbol{w}} s = p_\dagger \, \nabla_{\boldsymbol{w}} s = p_\dagger \, \boldsymbol{J}_s^\top = p_\dagger \, \boldsymbol{J}_z^\top \nabla_{\boldsymbol{z}} s = \boldsymbol{J}_z^\top (\boldsymbol{p} - \boldsymbol{y}), \tag{47}$$

where the last equality uses (46). Equivalently,

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = p_\dagger \big(\nabla_{\boldsymbol{w}} z_\dagger(\boldsymbol{w}) - \nabla_{\boldsymbol{w}} z_\star(\boldsymbol{w})\big). \tag{48}$$

Thus the scalar-margin representation preserves the softmax cross-entropy value and gradient exactly. The approximation introduced by FGN begins only at the Gauss-Newton curvature level.

## C Derivations for the FGN Curvature

This appendix collects the derivations and proofs deferred from Section 3. All identities below use the reordered logit coordinates $(z_\star, \boldsymbol{z}_{-\star})$ introduced in Section 2.

### C.1 Two exact Hessian decompositions

The exact Hessian can be written in two equivalent ways, corresponding to the multiclass logit map $\boldsymbol{z}(\boldsymbol{w})$ and the scalar-margin map $s(\boldsymbol{w})$.

First, view the loss as the composition $\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = L(\boldsymbol{z}(\boldsymbol{w}))$ with $L(\boldsymbol{z}) = -\log p_\star$. Since $\nabla_{\boldsymbol{z}} L = \boldsymbol{p} - \boldsymbol{y}$ and $\nabla^2_{\boldsymbol{z}} L = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top$, the chain rule gives

$$\nabla^2_{\boldsymbol{w}} \ell = \boldsymbol{J}_z^\top \big(\operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top\big) \boldsymbol{J}_z + \sum_{c=1}^{C} (p_c - y_c)\, \nabla^2_{\boldsymbol{w}} z_c. \tag{49}$$

Second, use the exact scalar form from Section 2, $\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \phi(s(\boldsymbol{w}))$, with $\phi'(s) = p_\dagger$ and $\phi''(s) = p_\star p_\dagger$. Applying the scalar chain rule gives

$$\nabla^2_{\boldsymbol{w}} \ell = \phi''(s)\, \nabla_{\boldsymbol{w}} s\, \nabla_{\boldsymbol{w}} s^\top + \phi'(s)\, \nabla^2_{\boldsymbol{w}} s = p_\star p_\dagger\, \boldsymbol{J}_s^\top \boldsymbol{J}_s + p_\dagger\, \nabla^2_{\boldsymbol{w}} s. \tag{50}$$

These identities clarify the hierarchy

$$\text{exact Hessian} \;\longrightarrow\; \text{multiclass GGN} \;\longrightarrow\; \text{FGN}.$$

The first arrow discards second derivatives of the logit map. The second arrow additionally removes part of the output-space curvature, as characterized in Theorem 3.1.

### C.2 Proof of Theorem 3.1

Recall the block form

$$\boldsymbol{p} = \begin{bmatrix} p_\star \\ \boldsymbol{p}_{-\star} \end{bmatrix}, \qquad\qquad \boldsymbol{p}_{-\star} = p_\dagger \boldsymbol{\rho}.$$

Expanding the softmax covariance in blocks gives

$$\operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top = \begin{bmatrix} p_\star - p_\star^2 & -p_\star \boldsymbol{p}_{-\star}^\top \\ -p_\star \boldsymbol{p}_{-\star} & \operatorname{diag}(\boldsymbol{p}_{-\star}) - \boldsymbol{p}_{-\star}\boldsymbol{p}_{-\star}^\top \end{bmatrix} = \begin{bmatrix} p_\star p_\dagger & -p_\star p_\dagger\, \boldsymbol{\rho}^\top \\ -p_\star p_\dagger\, \boldsymbol{\rho} & p_\dagger \operatorname{diag}(\boldsymbol{\rho}) - p_\dagger^2 \boldsymbol{\rho}\boldsymbol{\rho}^\top \end{bmatrix}. \tag{51}$$

Using $p_\dagger - p_\dagger^2 = p_\star p_\dagger$, the lower-right block becomes

$$p_\dagger \operatorname{diag}(\boldsymbol{\rho}) - p_\dagger^2 \boldsymbol{\rho}\boldsymbol{\rho}^\top = p_\star p_\dagger\, \boldsymbol{\rho}\boldsymbol{\rho}^\top + p_\dagger \big(\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top\big). \tag{52}$$

Substituting (52) into (51) yields

$$\operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top = p_\star p_\dagger \begin{bmatrix} 1 & -\boldsymbol{\rho}^\top \\ -\boldsymbol{\rho} & \boldsymbol{\rho}\boldsymbol{\rho}^\top \end{bmatrix} + p_\dagger \begin{bmatrix} 0 & 0 \\ 0 & \operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top \end{bmatrix}. \tag{53}$$

Since $\nabla_{\boldsymbol{z}} s = \begin{bmatrix} -1 \\ \boldsymbol{\rho} \end{bmatrix}$, the first matrix is exactly $\nabla_{\boldsymbol{z}} s\, \nabla_{\boldsymbol{z}} s^\top$, which gives (14). The residual is PSD since $\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top$ is the covariance matrix of a categorical random variable with probabilities $\boldsymbol{\rho}$.

For the parameter-space identity, multiply on the left and right by $\boldsymbol{J}_z^\top$ and $\boldsymbol{J}_z$:

$$\boldsymbol{J}_z^\top \big(\operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top\big) \boldsymbol{J}_z = p_\star p_\dagger\, \boldsymbol{J}_z^\top \nabla_{\boldsymbol{z}} s\, \nabla_{\boldsymbol{z}} s^\top \boldsymbol{J}_z + p_\dagger\, \boldsymbol{J}_{-\star}^\top \big(\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top\big) \boldsymbol{J}_{-\star}. \tag{54}$$

Using $\boldsymbol{J}_s = (\nabla_{\boldsymbol{z}} s)^\top \boldsymbol{J}_z$, the first term equals $p_\star p_\dagger\, \boldsymbol{J}_s^\top \boldsymbol{J}_s = \boldsymbol{H}^{\mathrm{FGN}}$, which yields (15).

### C.3 Proof of Proposition 3.2

We prove the items in order.

**Item 1.** Since $p_\star p_\dagger > 0$ for finite logits and $\boldsymbol{J}_s^\top \boldsymbol{J}_s \succeq 0$, we have $\boldsymbol{H}^{\mathrm{FGN}} \succeq 0$. The matrix $\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top$ is the covariance matrix of a categorical distribution with probabilities $\boldsymbol{\rho}$, and is therefore PSD. Hence the residual term in (15) is PSD, which implies

$$0 \preceq \boldsymbol{H}^{\mathrm{FGN}} \preceq \boldsymbol{H}^{\mathrm{GGN}}.$$

**Item 2.** If $C = 2$, then the competitor set has size one, so $\boldsymbol{\rho} = [1]$. Therefore $\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top = 0$, and (15) reduces to $\boldsymbol{H}^{\mathrm{FGN}} = \boldsymbol{H}^{\mathrm{GGN}}$.

**Item 3.** For any probability vector $\boldsymbol{\rho}$, the covariance matrix $\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top$ vanishes if and only if the distribution is degenerate, that is, if and only if $\boldsymbol{\rho}$ is one-hot. Hence the residual in (14) vanishes if and only if $\boldsymbol{\rho}$ is one-hot, proving the logit-space exactness statement.

Sufficiency for parameter-space exactness is immediate from (15). The converse need not hold in parameter space because multiplication by $\boldsymbol{J}_{-\star}$ can annihilate a nonzero logit-space residual. For example, if all competitor Jacobian rows are identical, then $\boldsymbol{J}_{-\star}$ has rank at most one and $\boldsymbol{J}_{-\star}^\top(\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top)\boldsymbol{J}_{-\star} = 0$ for every $\boldsymbol{\rho}$, even when $\boldsymbol{\rho}$ is diffuse.

**Item 4.** Let $\boldsymbol{J}_1, \dots, \boldsymbol{J}_{C-1}$ denote the rows of $\boldsymbol{J}_{-\star}$, and define $\bar{\boldsymbol{J}}_\rho \coloneqq \boldsymbol{\rho}^\top \boldsymbol{J}_{-\star} = \sum_{j=1}^{C-1} \rho_j \boldsymbol{J}_j$. Then

$$\boldsymbol{J}_{-\star}^\top\big(\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top\big)\boldsymbol{J}_{-\star} = \sum_{j=1}^{C-1} \rho_j\, \boldsymbol{J}_j^\top \boldsymbol{J}_j - \bar{\boldsymbol{J}}_\rho^\top \bar{\boldsymbol{J}}_\rho. \tag{55}$$

On the other hand,

$$\sum_{j=1}^{C-1} \rho_j\, (\boldsymbol{J}_j - \bar{\boldsymbol{J}}_\rho)^\top (\boldsymbol{J}_j - \bar{\boldsymbol{J}}_\rho) = \sum_{j=1}^{C-1} \rho_j\, \boldsymbol{J}_j^\top \boldsymbol{J}_j - \bar{\boldsymbol{J}}_\rho^\top \bar{\boldsymbol{J}}_\rho, \tag{56}$$

because $\sum_j \rho_j = 1$ and $\sum_j \rho_j \boldsymbol{J}_j = \bar{\boldsymbol{J}}_\rho$. Combining this identity with (15) proves (16).

**Diagnostic $\xi$.** The trace identity follows immediately:

$$\operatorname{Tr}\big(\operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top\big) = \sum_{j=1}^{C-1} \rho_j - \sum_{j=1}^{C-1} \rho_j^2 = 1 - \|\boldsymbol{\rho}\|_2^2 = \xi. \tag{57}$$

Because $\boldsymbol{\rho}$ is a probability vector on $C-1$ coordinates, $\|\boldsymbol{\rho}\|_2^2 \in [1/(C-1), 1]$, with the lower bound attained by the uniform distribution and the upper bound attained by one-hot distributions. Therefore

$$0 \le \xi \le 1 - \frac{1}{C-1}. \tag{58}$$

Finally, the dropped logit-space term in (14) has trace

$$\operatorname{Tr}\left(p_\dagger \begin{bmatrix} 0 & 0 \\ 0 & \operatorname{diag}(\boldsymbol{\rho}) - \boldsymbol{\rho}\boldsymbol{\rho}^\top \end{bmatrix}\right) = p_\dagger\, \xi. \tag{59}$$

Hence, for fixed $p_\dagger$, the trace of the logit-space mismatch is maximized by the most diffuse competitor distribution and vanishes when the competitor distribution is one-hot.

### C.4 Batch gradient and curvature identities

Let

$$\mathcal{L}(\boldsymbol{w}; \mathcal{B}) = \frac{1}{b} \sum_{i=1}^{b} \ell(\boldsymbol{w}; \boldsymbol{x}_i, \boldsymbol{y}_i).$$

For each example $i$, Section 2 gives the exact gradient

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}; \boldsymbol{x}_i, \boldsymbol{y}_i) = p_\dagger^{(i)}\, \boldsymbol{J}_{s,i}^\top, \qquad \boldsymbol{J}_{s,i}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} s_i(\boldsymbol{w})^\top.$$

Here the subscript $i$ indexes examples, not competitor classes. Averaging over the batch yields

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}; \mathcal{B}) = \frac{1}{b} \sum_{i=1}^{b} p_\dagger^{(i)}\, \boldsymbol{J}_{s,i}^\top = \frac{1}{b}\, \boldsymbol{J}(\boldsymbol{w})^\top \boldsymbol{r}(\boldsymbol{w}), \tag{60}$$

where $r_i(\boldsymbol{w}) = p_\dagger^{(i)}(\boldsymbol{w})$ and $\boldsymbol{J}(\boldsymbol{w})$ stacks the rows $\boldsymbol{J}_{s,i}(\boldsymbol{w})$.

Likewise, the single-example FGN curvature is

$$\boldsymbol{H}_i^{\mathrm{FGN}}(\boldsymbol{w}) = p_\star^{(i)} p_\dagger^{(i)}\, \boldsymbol{J}_{s,i}^\top \boldsymbol{J}_{s,i}.$$

Averaging again gives

$$\boldsymbol{H}^{\mathrm{FGN}}(\boldsymbol{w};\mathcal{B}) = \frac{1}{b}\sum_{i=1}^{b} p_\star^{(i)} p_\dagger^{(i)}\, \boldsymbol{J}_{s,i}^\top \boldsymbol{J}_{s,i} = \frac{1}{b}\,\boldsymbol{J}(\boldsymbol{w})^\top \boldsymbol{Q}(\boldsymbol{w})\,\boldsymbol{J}(\boldsymbol{w}), \tag{61}$$

where $\boldsymbol{Q}(\boldsymbol{w}) = \mathrm{diag}(q_1(\boldsymbol{w}), \dots, q_b(\boldsymbol{w}))$ with $q_i(\boldsymbol{w}) = p_\star^{(i)}(\boldsymbol{w}) p_\dagger^{(i)}(\boldsymbol{w})$. For finite logits, each $q_i(\boldsymbol{w}) \in (0, 1/4]$, so $\boldsymbol{Q}(\boldsymbol{w})$ is diagonal and strictly positive.

# D Derivations for the FGN Solver

This appendix establishes the equivalence between the parameter-space and row-space systems, gives the computational and storage accounting, relates row-space residuals to parameter-space residuals, and provides a simplified convergence analysis for the preconditioned update. The equivalence and residual identities use the shared-batch quantities from Section 4.1, whereas the convergence analysis in Appendices D.3 and D.5 assumes independently sampled gradient and curvature mini-batches.

## D.1 Equivalence of parameter- and row-space solves

**Lemma D.1.** *Assume $\lambda > 0$. Then the damped parameter-space system* (20)*, the unwhitened row-space system* (23)*, and the whitened row-space system* (25) *are equivalent via the representations*

$$\boldsymbol{d} = -\boldsymbol{J}^\top \boldsymbol{\delta} = -\tilde{\boldsymbol{J}}^\top \boldsymbol{u}. \tag{62}$$

*In particular, solving* (25) *and backprojecting with* (26) *returns exactly the unique solution of* (20)*.*

*Proof.* Recall the definitions

$$\boldsymbol{g} = \frac{1}{b}\,\boldsymbol{J}^\top \boldsymbol{r}, \qquad \boldsymbol{H}^{\mathrm{FGN}} = \frac{1}{b}\,\boldsymbol{J}^\top \boldsymbol{Q}\boldsymbol{J}, \tag{63}$$

$$\tilde{\boldsymbol{J}} \coloneqq \boldsymbol{Q}^{1/2}\boldsymbol{J}, \qquad \tilde{\boldsymbol{r}} \coloneqq \boldsymbol{Q}^{-1/2}\boldsymbol{r}, \qquad \boldsymbol{K} \coloneqq \tilde{\boldsymbol{J}}\tilde{\boldsymbol{J}}^\top. \tag{64}$$

Let

$$\boldsymbol{A} \coloneqq \boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d = \frac{1}{b}\,\boldsymbol{J}^\top \boldsymbol{Q}\boldsymbol{J} + \lambda \boldsymbol{I}_d. \tag{65}$$

Because $\boldsymbol{Q} \succ 0$, we have $\boldsymbol{J}^\top \boldsymbol{Q}\boldsymbol{J} \succeq 0$, and since $\lambda > 0$, the parameter-space operator $\boldsymbol{A}$ is symmetric positive definite. Hence the parameter-space system has a unique solution.

The whitened row-space operator

$$\boldsymbol{K} + b\,\lambda \boldsymbol{I}_b \tag{66}$$

is also symmetric positive definite, so the whitened row-space system has a unique solution. Moreover,

$$\boldsymbol{Q}\boldsymbol{J}\boldsymbol{J}^\top + b\,\lambda \boldsymbol{I}_b = \boldsymbol{Q}^{1/2}\big(\boldsymbol{K} + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{Q}^{-1/2}, \tag{67}$$

so the unwhitened row-space operator is similar to the SPD matrix $\boldsymbol{K} + b\,\lambda \boldsymbol{I}_b$ and is therefore invertible.

If $\boldsymbol{\delta}$ solves the unwhitened row-space system

$$\big(\boldsymbol{Q}\boldsymbol{J}\boldsymbol{J}^\top + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{\delta} = \boldsymbol{r} \tag{68}$$

and we set

$$\boldsymbol{d} \coloneqq -\boldsymbol{J}^\top \boldsymbol{\delta}, \tag{69}$$

then

$$\begin{aligned}\big(\boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)\boldsymbol{d} &= \left(\frac{1}{b}\,\boldsymbol{J}^\top \boldsymbol{Q}\boldsymbol{J} + \lambda \boldsymbol{I}_d\right)(-\boldsymbol{J}^\top \boldsymbol{\delta}) \\ &= -\frac{1}{b}\,\boldsymbol{J}^\top \big(\boldsymbol{Q}\boldsymbol{J}\boldsymbol{J}^\top + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{\delta} = -\frac{1}{b}\,\boldsymbol{J}^\top \boldsymbol{r} = -\boldsymbol{g}.\end{aligned} \tag{70}$$

So the backprojected unwhitened row-space solution solves the parameter-space system.

If $\boldsymbol{u}$ solves the whitened row-space system

$$\big(\boldsymbol{K} + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{u} = \tilde{\boldsymbol{r}} \tag{71}$$

and we set

$$\boldsymbol{d} \coloneqq -\tilde{\boldsymbol{J}}^\top \boldsymbol{u}, \tag{72}$$

then

$$\begin{aligned}\big(\boldsymbol{H}^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)\boldsymbol{d} &= \left(\frac{1}{b}\,\tilde{\boldsymbol{J}}^\top \tilde{\boldsymbol{J}} + \lambda \boldsymbol{I}_d\right)(-\tilde{\boldsymbol{J}}^\top \boldsymbol{u}) \\ &= -\frac{1}{b}\,\tilde{\boldsymbol{J}}^\top\big(\tilde{\boldsymbol{J}}\tilde{\boldsymbol{J}}^\top + b\,\lambda \boldsymbol{I}_b\big)\boldsymbol{u} = -\frac{1}{b}\,\tilde{\boldsymbol{J}}^\top \tilde{\boldsymbol{r}} = -\boldsymbol{g}.\end{aligned} \tag{73}$$

So the backprojected whitened row-space solution also solves the same parameter-space system.

Conversely, let $\boldsymbol{d}$ be the unique solution of the parameter-space system. Let $\boldsymbol{\delta}$ be the unique solution of the unwhitened row-space system. By the implication above, $\boldsymbol{d}' \coloneqq -\boldsymbol{J}^\top \boldsymbol{\delta}$ also solves the parameter-space system. Uniqueness therefore implies $\boldsymbol{d}' = \boldsymbol{d}$. Finally, the linear change of variables

$$\boldsymbol{u} = \boldsymbol{Q}^{-1/2}\boldsymbol{\delta}, \qquad \boldsymbol{\delta} = \boldsymbol{Q}^{1/2}\boldsymbol{u}, \tag{74}$$

is invertible, so the unwhitened and whitened row-space systems are equivalent and yield the same backprojected direction. □

### D.2 Cost accounting

Recall the whitened row-space operator

$$\boldsymbol{B}\boldsymbol{v} = \tilde{\boldsymbol{J}}\big(\tilde{\boldsymbol{J}}^\top \boldsymbol{v}\big) + b\lambda \boldsymbol{v}, \qquad \tilde{\boldsymbol{J}} = \boldsymbol{Q}^{1/2}\boldsymbol{J}, \qquad \boldsymbol{v} \in \mathbb{R}^b. \tag{75}$$

A matrix-free application proceeds as

$$\boldsymbol{a} = \boldsymbol{Q}^{1/2}\boldsymbol{v}, \qquad \boldsymbol{\beta} = \boldsymbol{J}^\top \boldsymbol{a}, \qquad \boldsymbol{\gamma} = \boldsymbol{J}\boldsymbol{\beta}, \qquad \boldsymbol{B}\boldsymbol{v} = \boldsymbol{Q}^{1/2}\boldsymbol{\gamma} + b\lambda \boldsymbol{v}. \tag{76}$$

Thus one application of $\boldsymbol{B}$ requires one vector-Jacobian product and one Jacobian-vector product with the batched true-vs-rest margin map, plus $\mathcal{O}(b)$ vector operations for diagonal scalings and damping.

In autodiff implementations, the batched margin map can be linearized once at the current parameters, and the resulting JVP/VJP closures can be reused across CG iterations.

For standard row-space CG with zero initialization and at most $N_{\mathrm{CG}}$ iterations, the additional solve work is bounded by

$$\begin{aligned}\mathrm{cost}(\text{extra FGN solve}) \le \mathrm{cost}(\text{linearize}) + N_{\mathrm{CG}}\Big[\mathrm{cost}(\mathrm{JVP}_s) + \mathrm{cost}(\mathrm{VJP}_s)\Big] \\ + \mathrm{cost}(\mathrm{VJP}_s) + \mathcal{O}\big((N_{\mathrm{CG}} + 1)b\big),\end{aligned} \tag{77}$$

where the subscript $s$ denotes the batched scalar-margin map, and the final VJP is the backprojection $\boldsymbol{d} = -\tilde{\boldsymbol{J}}^\top \boldsymbol{u}$. The linearization term includes the one-time setup of reusable derivative routines. For example, `jax.linearize` in JAX [Bradbury et al., 2018] evaluates the margin map at the current parameters and returns a reusable JVP function. The costs of linearization, JVPs, and VJPs depend on the model and implementation.

The row-space system has dimension $b$, and no $d \times d$ matrix is formed. A dense direct row-space solve would require explicitly forming $\boldsymbol{K} + b\lambda \boldsymbol{I}_b$ and paying $\mathcal{O}(b^3)$ for a factorization. In the generic matrix-free setting, explicitly forming $\boldsymbol{K}$ itself would require probing the row-space operator with many right-hand sides. Matrix-free CG instead uses only repeated applications of (27).

Table 1 summarizes derivative setup, gradient and solve operations, and persistent optimizer vectors. The final $\mathcal{O}(d)$ parameter update is common to all methods. The FGN row describes standard matrix-free row-space CG. For full GGN, we consider an SGN implementation [Gargiani et al., 2020] using parameter-space CG. The CG counts assume zero initialization and no additional preconditioning.

Table 1: Computational and storage comparison of FGN, full GGN, and first-order methods.

| Method | Solve size | Core operations | Persistent vectors |
|---|---|---|---|
| FGN | $b$ | Margin linearization; at most $N_{\mathrm{CG}}$ margin JVP/VJP pairs; $\mathcal{O}(N_{\mathrm{CG}}b)$ row-space vector work; one final VJP | $\mathcal{O}(b)$ CG vectors |
| Full GGN | $d$ | Logit derivative setup and gradient evaluation; at most $N_{\mathrm{CG}}$ logit JVP/VJP pairs; $\mathcal{O}(N_{\mathrm{CG}}bC)$ covariance work; $\mathcal{O}(N_{\mathrm{CG}}d)$ CG vector work | $\mathcal{O}(d)$ CG vectors |
| SGD | None | Gradient evaluation | None; $\mathcal{O}(d)$ with momentum |
| Adam | None | Gradient evaluation and $\mathcal{O}(d)$ moment updates | Two $\mathcal{O}(d)$ moment vectors |

FGN forms $\tilde{\boldsymbol{r}} = \boldsymbol{Q}^{-1/2}\boldsymbol{r}$ directly from the probabilities, so it does not require a separate parameter-space gradient evaluation before CG. For full GGN, each logit JVP produces a $b \times C$ perturbation. For each example, multiplication by the softmax covariance is

$$\boldsymbol{v} \longmapsto \boldsymbol{p} \odot \boldsymbol{v} - \boldsymbol{p}(\boldsymbol{p}^\top \boldsymbol{v}), \qquad \boldsymbol{v} \in \mathbb{R}^C,$$

where $\odot$ denotes elementwise multiplication. These multiplications cost $\mathcal{O}(C)$ per example, or $\mathcal{O}(bC)$ per mini-batch. A logit VJP then maps the result back to parameter space. No dense $C \times C$ covariance matrix is formed.

The persistent vector counts include only CG vectors or optimizer moment vectors; they exclude model parameters, gradients, autodiff temporaries, and stored linearizations. They therefore do not represent peak training memory. FGN still uses parameter-sized intermediate vectors, and its linearization can retain model activations. Both second-order methods also evaluate all $C$ logits and probabilities. CG convergence depends on the spectrum and right-hand side, so the smaller FGN system does not by itself imply fewer iterations.

**Lifted row-space CG.** Let $\boldsymbol{p}_k$ be the CG search direction and $a_k$ its step length. Each operator application already computes $\boldsymbol{q}_k = \tilde{\boldsymbol{J}}^\top \boldsymbol{p}_k$. Starting from $\boldsymbol{d}_0 = \boldsymbol{0}$, the lifted variant accumulates

$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k - a_k \boldsymbol{q}_k.$$

Since $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + a_k \boldsymbol{p}_k$, we have $\boldsymbol{d}_k = -\tilde{\boldsymbol{J}}^\top \boldsymbol{u}_k$ in exact arithmetic. Thus lifting eliminates the final VJP at the cost of storing a $d$-dimensional accumulator updated at each iteration.

### D.3 A simplified stochastic preconditioned viewpoint

We record one standard convergence template for the solver interpretation from Section 4.3. It applies only to a simplified variant with fixed damping and independent curvature and gradient mini-batches.

Let $\mathcal{D}$ be a finite training set and define the empirical risk

$$\mathcal{R}(\boldsymbol{w}) \coloneqq \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D}} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}). \tag{78}$$

For an i.i.d. mini-batch $\mathcal{B}$ of size $b$, the batch loss $\mathcal{L}(\boldsymbol{w}; \mathcal{B})$ satisfies

$$\mathbb{E}_{\mathcal{B}}\big[\mathcal{L}(\boldsymbol{w}; \mathcal{B})\big] = \mathcal{R}(\boldsymbol{w}). \tag{79}$$

At iteration $t$, let $\hat{\mathcal{B}}_t$ be an i.i.d. curvature mini-batch and $\mathcal{B}_t$ an independent i.i.d. gradient mini-batch. We form

$$\boldsymbol{H}_t^{\mathrm{FGN}} \coloneqq \boldsymbol{H}^{\mathrm{FGN}}(\boldsymbol{w}_t; \hat{\mathcal{B}}_t), \qquad \boldsymbol{g}_t \coloneqq \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}_t; \mathcal{B}_t), \tag{80}$$

fix a damping parameter $\lambda > 0$, define

$$\boldsymbol{P}_t \coloneqq \big(\boldsymbol{H}_t^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d\big)^{-1}, \tag{81}$$

and update

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha_t \boldsymbol{P}_t \boldsymbol{g}_t. \tag{82}$$

Let $\mathcal{F}_t$ denote the sigma-algebra generated by $\boldsymbol{w}_0, \hat{\mathcal{B}}_0, \mathcal{B}_0, \ldots, \hat{\mathcal{B}}_{t-1}, \mathcal{B}_{t-1}, \hat{\mathcal{B}}_t$. Then $\boldsymbol{P}_t$ is $\mathcal{F}_t$-measurable and $\mathcal{B}_t$ is independent of $\mathcal{F}_t$.

**Assumption D.2** (Smooth risk)**.** The risk $\mathcal{R}$ is bounded below and has $L_g$-Lipschitz gradient: for all $\boldsymbol{w}, \boldsymbol{u}$,

$$\|\nabla\mathcal{R}(\boldsymbol{w}) - \nabla\mathcal{R}(\boldsymbol{u})\| \le L_g\|\boldsymbol{w} - \boldsymbol{u}\|. \tag{83}$$

**Assumption D.3** (Unbiased stochastic gradients)**.** For all $t$,

$$\mathbb{E}[\boldsymbol{g}_t \mid \mathcal{F}_t] = \nabla\mathcal{R}(\boldsymbol{w}_t), \qquad \mathbb{E}[\|\boldsymbol{g}_t\|^2 \mid \mathcal{F}_t] \le G^2 \tag{84}$$

almost surely, for some constant $G < \infty$.

**Assumption D.4** (Uniform spectral bounds)**.** There exist constants $0 < \underline{\kappa} \le \overline{\kappa} < \infty$ such that, almost surely for all $t$,

$$\underline{\kappa}\,\boldsymbol{I}_d \preceq \boldsymbol{P}_t \preceq \overline{\kappa}\,\boldsymbol{I}_d. \tag{85}$$

**Assumption D.5** (Step sizes)**.** The step sizes are positive and satisfy

$$\sum_{t=0}^{\infty}\alpha_t = \infty, \qquad \sum_{t=0}^{\infty}\alpha_t^2 < \infty. \tag{86}$$

**Theorem D.6** (A simplified stochastic FGN guarantee)**.** *Suppose Assumptions D.2–D.5 hold, and let* $\{\boldsymbol{w}_t\}$ *be generated by* (82)*. Then:*

1. *the sequence* $\{\mathcal{R}(\boldsymbol{w}_t)\}$ *converges almost surely to a finite random variable;*
2. $$\sum_{t=0}^{\infty}\alpha_t\,\mathbb{E}\big[\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2\big] < \infty; \tag{87}$$
3. *in particular,* $$\liminf_{t\to\infty}\mathbb{E}\big[\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2\big] = 0. \tag{88}$$

*Proof.* By $L_g$-smoothness of $\mathcal{R}$,

$$\mathcal{R}(\boldsymbol{w}_{t+1}) \le \mathcal{R}(\boldsymbol{w}_t) + \nabla\mathcal{R}(\boldsymbol{w}_t)^\top(\boldsymbol{w}_{t+1} - \boldsymbol{w}_t) + \frac{L_g}{2}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2. \tag{89}$$

Substituting $\boldsymbol{w}_{t+1} - \boldsymbol{w}_t = -\alpha_t\boldsymbol{P}_t\boldsymbol{g}_t$ gives

$$\mathcal{R}(\boldsymbol{w}_{t+1}) \le \mathcal{R}(\boldsymbol{w}_t) - \alpha_t\nabla\mathcal{R}(\boldsymbol{w}_t)^\top\boldsymbol{P}_t\boldsymbol{g}_t + \frac{L_g\alpha_t^2}{2}\|\boldsymbol{P}_t\boldsymbol{g}_t\|^2. \tag{90}$$

Taking conditional expectation given $\mathcal{F}_t$ and using Assumption D.3 yields

$$\mathbb{E}\big[\mathcal{R}(\boldsymbol{w}_{t+1}) \mid \mathcal{F}_t\big] \le \mathcal{R}(\boldsymbol{w}_t) - \alpha_t\nabla\mathcal{R}(\boldsymbol{w}_t)^\top\boldsymbol{P}_t\nabla\mathcal{R}(\boldsymbol{w}_t) + \frac{L_g\alpha_t^2}{2}\mathbb{E}\big[\|\boldsymbol{P}_t\boldsymbol{g}_t\|^2 \mid \mathcal{F}_t\big]. \tag{91}$$

By Assumption D.4,

$$\nabla\mathcal{R}(\boldsymbol{w}_t)^\top\boldsymbol{P}_t\nabla\mathcal{R}(\boldsymbol{w}_t) \ge \underline{\kappa}\,\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2, \tag{92}$$

$$\|\boldsymbol{P}_t\boldsymbol{g}_t\|^2 \le \overline{\kappa}^2\|\boldsymbol{g}_t\|^2. \tag{93}$$

Using Assumption D.3 again,

$$\mathbb{E}\big[\mathcal{R}(\boldsymbol{w}_{t+1}) \mid \mathcal{F}_t\big] \le \mathcal{R}(\boldsymbol{w}_t) - \underline{\kappa}\,\alpha_t\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2 + \frac{L_g\overline{\kappa}^2G^2}{2}\,\alpha_t^2. \tag{94}$$

This is an almost-supermartingale recursion with summable error term because $\sum_t \alpha_t^2 < \infty$. Applying the Robbins-Siegmund theorem [Robbins and Siegmund, 1971] to the nonnegative sequence $\mathcal{R}(\boldsymbol{w}_t) - \inf_{\boldsymbol{w}}\mathcal{R}(\boldsymbol{w})$, we obtain that $\{\mathcal{R}(\boldsymbol{w}_t)\}$ converges almost surely to a finite limit and

$$\sum_{t=0}^{\infty}\alpha_t\,\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2 < \infty \qquad \text{almost surely.} \tag{95}$$

Taking full expectations in (94) and summing over $t$ also gives

$$\sum_{t=0}^{\infty}\alpha_t\,\mathbb{E}\big[\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2\big] < \infty. \tag{96}$$

Since $\sum_t \alpha_t = \infty$ and each term is nonnegative, this implies

$$\liminf_{t\to\infty}\mathbb{E}\big[\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2\big] = 0. \tag{97}$$

□

### D.4 Residual transfer from row space to parameter space

For $\lambda_t > 0$, let $\boldsymbol{u}_t \in \mathbb{R}^b$ be any row-space iterate and define

$$\boldsymbol{e}_t \coloneqq \big(\boldsymbol{K}_t + b\,\lambda_t \boldsymbol{I}_b\big)\boldsymbol{u}_t - \tilde{\boldsymbol{r}}_t, \qquad\qquad \boldsymbol{d}_t \coloneqq -\tilde{\boldsymbol{J}}_t^\top \boldsymbol{u}_t. \tag{98}$$

Then

$$\begin{aligned}\big(\boldsymbol{H}_t^{\mathrm{FGN}} + \lambda_t \boldsymbol{I}_d\big)\boldsymbol{d}_t + \boldsymbol{g}_t &= \left(\frac{1}{b}\,\tilde{\boldsymbol{J}}_t^\top \tilde{\boldsymbol{J}}_t + \lambda_t \boldsymbol{I}_d\right)(-\tilde{\boldsymbol{J}}_t^\top \boldsymbol{u}_t) + \frac{1}{b}\,\tilde{\boldsymbol{J}}_t^\top \tilde{\boldsymbol{r}}_t \\ &= -\frac{1}{b}\,\tilde{\boldsymbol{J}}_t^\top\big(\tilde{\boldsymbol{J}}_t \tilde{\boldsymbol{J}}_t^\top + b\,\lambda_t \boldsymbol{I}_b\big)\boldsymbol{u}_t + \frac{1}{b}\,\tilde{\boldsymbol{J}}_t^\top \tilde{\boldsymbol{r}}_t = -\frac{1}{b}\,\tilde{\boldsymbol{J}}_t^\top \boldsymbol{e}_t. \end{aligned} \tag{99}$$

It is therefore natural to define the parameter-space residual by

$$\boldsymbol{\varepsilon}_t \coloneqq \big(\boldsymbol{H}_t^{\mathrm{FGN}} + \lambda_t \boldsymbol{I}_d\big)\boldsymbol{d}_t + \boldsymbol{g}_t = -\frac{1}{b}\,\tilde{\boldsymbol{J}}_t^\top \boldsymbol{e}_t. \tag{100}$$

Taking Euclidean norms and using the operator norm gives

$$\|\boldsymbol{\varepsilon}_t\| = \frac{1}{b}\,\|\tilde{\boldsymbol{J}}_t^\top \boldsymbol{e}_t\| \le \frac{1}{b}\,\|\tilde{\boldsymbol{J}}_t^\top\|\,\|\boldsymbol{e}_t\| = \frac{1}{b}\,\|\tilde{\boldsymbol{J}}_t\|\,\|\boldsymbol{e}_t\|. \tag{101}$$

Hence a small row-space residual implies a small parameter-space residual, up to the multiplicative factor $\|\tilde{\boldsymbol{J}}_t\|/b$.

This bound is the basic link between the row-space residual monitored by CG and parameter-space accuracy. However, a relative row-space stopping rule of the form $\|\boldsymbol{e}_t\| \le \tau_t \|\tilde{\boldsymbol{r}}_t\|$ does not by itself imply the relative parameter-space residual condition used in Lemma D.7 without additional control of $\|\tilde{\boldsymbol{J}}_t\|$ and $\|\boldsymbol{g}_t\|$. For that reason, the inexact solve lemma states the parameter-space residual condition explicitly.

### D.5 Inexact linear solves

We return to the independent-batch update in Theorem D.6 and allow its parameter-space system to be solved approximately. The following lemma gives sufficient conditions in terms of the parameter-space residual.

**Lemma D.7** (Inexact solves under a relative residual condition)**.** *Assume Assumptions D.2–D.5 hold, with the same filtration and mini-batch independence as in Theorem D.6. Let*

$$\boldsymbol{A}_t \coloneqq \boldsymbol{H}_t^{\mathrm{FGN}} + \lambda \boldsymbol{I}_d, \qquad \boldsymbol{P}_t \coloneqq \boldsymbol{A}_t^{-1}, \tag{102}$$

*and consider updates*

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t \boldsymbol{d}_t, \tag{103}$$

*where $\boldsymbol{d}_t$ is produced by an iterative solver applied to $\boldsymbol{A}_t \boldsymbol{d} = -\boldsymbol{g}_t$. Define the parameter-space residual*

$$\boldsymbol{\varepsilon}_t \coloneqq \boldsymbol{A}_t \boldsymbol{d}_t + \boldsymbol{g}_t. \tag{104}$$

*Assume that for some deterministic nonnegative sequence $\{\eta_t\}$,*

$$\|\boldsymbol{\varepsilon}_t\| \le \eta_t \|\boldsymbol{g}_t\| \qquad \textit{almost surely,} \tag{105}$$

*with*

$$\sup_t \eta_t < \infty, \qquad \sum_{t=0}^{\infty} \alpha_t \eta_t^2 < \infty. \tag{106}$$

*Then the conclusions of Theorem D.6 remain valid.*

*Proof.* Since $\boldsymbol{\varepsilon}_t = \boldsymbol{A}_t \boldsymbol{d}_t + \boldsymbol{g}_t$ and $\boldsymbol{P}_t = \boldsymbol{A}_t^{-1}$, the inexact direction can be written as

$$\boldsymbol{d}_t = -\boldsymbol{P}_t \boldsymbol{g}_t + \boldsymbol{P}_t \boldsymbol{\varepsilon}_t. \tag{107}$$

Using smoothness of $\mathcal{R}$,

$$\mathcal{R}(\boldsymbol{w}_{t+1}) \le \mathcal{R}(\boldsymbol{w}_t) + \alpha_t \nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{d}_t + \frac{L_g \alpha_t^2}{2}\|\boldsymbol{d}_t\|^2. \tag{108}$$

Substituting (107) and taking conditional expectation given $\mathcal{F}_t$ gives

$$\begin{aligned}\mathbb{E}\big[\mathcal{R}(\boldsymbol{w}_{t+1}) \mid \mathcal{F}_t\big] \le \mathcal{R}(\boldsymbol{w}_t) &- \alpha_t \nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \nabla\mathcal{R}(\boldsymbol{w}_t) \\ &+ \alpha_t \mathbb{E}\big[\nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \boldsymbol{\varepsilon}_t \mid \mathcal{F}_t\big] + \frac{L_g \alpha_t^2}{2}\mathbb{E}\big[\|\boldsymbol{d}_t\|^2 \mid \mathcal{F}_t\big],\end{aligned} \tag{109}$$

where we used $\mathbb{E}[\boldsymbol{g}_t \mid \mathcal{F}_t] = \nabla\mathcal{R}(\boldsymbol{w}_t)$ and the fact that $\boldsymbol{P}_t$ is $\mathcal{F}_t$-measurable.

For each realization, Cauchy-Schwarz in the $\boldsymbol{P}_t$ metric and Young's inequality give

$$\begin{aligned}\nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \boldsymbol{\varepsilon}_t &\le \frac{1}{2}\nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \nabla\mathcal{R}(\boldsymbol{w}_t) + \frac{1}{2}\boldsymbol{\varepsilon}_t^\top \boldsymbol{P}_t \boldsymbol{\varepsilon}_t \\ &\le \frac{1}{2}\nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \nabla\mathcal{R}(\boldsymbol{w}_t) + \frac{\overline{\kappa}}{2}\eta_t^2 \|\boldsymbol{g}_t\|^2.\end{aligned} \tag{110}$$

Therefore

$$\mathbb{E}\big[\nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \boldsymbol{\varepsilon}_t \mid \mathcal{F}_t\big] \le \frac{1}{2}\nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \nabla\mathcal{R}(\boldsymbol{w}_t) + \frac{\overline{\kappa}}{2}\eta_t^2 \mathbb{E}[\|\boldsymbol{g}_t\|^2 \mid \mathcal{F}_t]. \tag{111}$$

Similarly, using $\boldsymbol{d}_t = -\boldsymbol{P}_t \boldsymbol{g}_t + \boldsymbol{P}_t \boldsymbol{\varepsilon}_t$,

$$\|\boldsymbol{d}_t\|^2 \le 2\|\boldsymbol{P}_t \boldsymbol{g}_t\|^2 + 2\|\boldsymbol{P}_t \boldsymbol{\varepsilon}_t\|^2 \le 2\overline{\kappa}^2(1+\eta_t^2)\|\boldsymbol{g}_t\|^2, \tag{112}$$

and hence

$$\mathbb{E}[\|\boldsymbol{d}_t\|^2 \mid \mathcal{F}_t] \le 2\overline{\kappa}^2(1+\eta_t^2)\mathbb{E}[\|\boldsymbol{g}_t\|^2 \mid \mathcal{F}_t]. \tag{113}$$

Substituting the last two bounds gives

$$\begin{aligned}\mathbb{E}\big[\mathcal{R}(\boldsymbol{w}_{t+1}) \mid \mathcal{F}_t\big] \le \mathcal{R}(\boldsymbol{w}_t) &- \frac{1}{2}\alpha_t \nabla\mathcal{R}(\boldsymbol{w}_t)^\top \boldsymbol{P}_t \nabla\mathcal{R}(\boldsymbol{w}_t) + \frac{\overline{\kappa}}{2}\alpha_t \eta_t^2 \mathbb{E}[\|\boldsymbol{g}_t\|^2 \mid \mathcal{F}_t] \\ &+ L_g \overline{\kappa}^2 \alpha_t^2 (1+\eta_t^2)\mathbb{E}[\|\boldsymbol{g}_t\|^2 \mid \mathcal{F}_t].\end{aligned} \tag{114}$$

Using Assumptions D.3 and D.4,

$$\mathbb{E}\big[\mathcal{R}(\boldsymbol{w}_{t+1}) \mid \mathcal{F}_t\big] \le \mathcal{R}(\boldsymbol{w}_t) - \frac{\underline{\kappa}}{2}\alpha_t \|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2 + \frac{\overline{\kappa} G^2}{2}\alpha_t \eta_t^2 + L_g \overline{\kappa}^2 G^2 \alpha_t^2 (1+\eta_t^2). \tag{115}$$

Because $\sup_t \eta_t < \infty$, $\sum_t \alpha_t^2 < \infty$, and $\sum_t \alpha_t \eta_t^2 < \infty$, the error terms in (115) are summable. The same Robbins-Siegmund argument as in the exact-solve case yields almost sure convergence of $\{\mathcal{R}(\boldsymbol{w}_t)\}$ and

$$\sum_{t=0}^{\infty} \alpha_t \|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2 < \infty \qquad \text{almost surely.}$$

Taking full expectations in (115) and summing over $t$ also gives

$$\sum_{t=0}^{\infty} \alpha_t \, \mathbb{E}\big[\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2\big] < \infty.$$

Since $\sum_t \alpha_t = \infty$ and the terms are nonnegative,

$$\liminf_{t\to\infty} \mathbb{E}\big[\|\nabla\mathcal{R}(\boldsymbol{w}_t)\|^2\big] = 0.$$

□

# E Algorithms

Algorithm 1 summarizes the core damped FGN step analyzed in Section 4.

Momentum, adaptive damping, preconditioning, CG warm starts, line search, and curvature reuse are implementation choices rather than part of the core equivalence established in Section 4.1.

**Algorithm 1** Core Fast Gauss-Newton (FGN) step

**Input:** mini-batch $\mathcal{B}_t$, parameters $\boldsymbol{w}_t$, damping $\lambda_t > 0$, step size $\alpha_t$, relative CG tolerance $\epsilon$, maximum CG iterations $N_{\mathrm{CG}}$.

**Output:** updated parameters $\boldsymbol{w}_{t+1}$.

1: Compute logits, softmax probabilities, margins $s_i = z_\dagger^{(i)} - z_\star^{(i)}$, and weights $q_i = p_\star^{(i)} p_\dagger^{(i)}$ for $i = 1, \dots, b$.

2: Define

$$\boldsymbol{Q}_t \coloneqq \mathrm{diag}(q_1, \dots, q_b), \qquad (\tilde{\boldsymbol{r}}_t)_i \coloneqq \sqrt{\frac{p_\dagger^{(i)}}{p_\star^{(i)}}}.$$

3: Linearize the margin map at $\boldsymbol{w}_t$ and construct its matrix-free JVP/VJP routines. Define the whitened Jacobian operator

$$\tilde{\boldsymbol{J}}_t \coloneqq \boldsymbol{Q}_t^{1/2} \boldsymbol{J}_t$$

with $\boldsymbol{Q}_t$ held fixed during JVP/VJP evaluations.

4: Define the row-space operator

$$\boldsymbol{B}_t \boldsymbol{v} \coloneqq \tilde{\boldsymbol{J}}_t \tilde{\boldsymbol{J}}_t^\top \boldsymbol{v} + b\lambda_t \boldsymbol{v}, \qquad \boldsymbol{v} \in \mathbb{R}^b.$$

5: Starting from $\boldsymbol{u} = \boldsymbol{0}$, apply CG to

$$\boldsymbol{B}_t \boldsymbol{u} = \tilde{\boldsymbol{r}}_t,$$

stopping when $\|\boldsymbol{B}_t \boldsymbol{u} - \tilde{\boldsymbol{r}}_t\| \le \epsilon \|\tilde{\boldsymbol{r}}_t\|$ or after $N_{\mathrm{CG}}$ iterations, and set the final iterate to $\boldsymbol{u}_t$.

6: Recover the parameter-space direction

$$\boldsymbol{d}_t \coloneqq -\tilde{\boldsymbol{J}}_t^\top \boldsymbol{u}_t.$$

7: Update

$$\boldsymbol{w}_{t+1} \coloneqq \boldsymbol{w}_t + \alpha_t \boldsymbol{d}_t.$$

# F Experiment Details

This appendix provides the experimental setups, hyperparameters, implementation details, and reporting protocols for the three experiments in Section 5. It also reports supporting diagnostics on competitor dispersion, optimizer steps, and CG budget sensitivity. All experiments were run on a single NVIDIA RTX A4000 GPU with 16 GB of memory.

## F.1 Mechanism checks

The three panels examine FGN from different angles: update cost, retained and discarded curvature, and the effect of damping on the resulting steps.

**Panel (a): optimizer update timing.** We use an affine softmax head on fixed Gaussian features. For each example,

$$\boldsymbol{z}_i = \boldsymbol{W}^\top \boldsymbol{x}_i + \boldsymbol{b}, \qquad \boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{2048}), \qquad \boldsymbol{W} \in \mathbb{R}^{2048 \times C}, \qquad \boldsymbol{b} \in \mathbb{R}^C.$$

Equivalently, for a batch,

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{1}\boldsymbol{b}^\top, \qquad \boldsymbol{X} \in \mathbb{R}^{b \times 2048}.$$

We sweep

$$C \in \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096\},$$

using batch size $512$ and $32768$ fixed synthetic examples with independently sampled uniform class labels. FGN uses row-space CG, whereas SGN uses parameter-space CG. Both second-order methods use fixed damping $\lambda = 1$, at most five CG iterations, tolerance $10^{-5}$, and no warm starts.

For each method and class count, we perform $128$ warmup updates, followed by $640$ timed updates in ten synchronized blocks of $64$. We report the mean and $\pm 1$ standard deviation over five seeds, using the median per-update block time from each run. The timings cover complete optimizer updates, including all logits, and exclude data generation, loading, compilation, and logging.

**Panel (b): output-space trace decomposition.** We fix $p_\star = 0.6$, set $p_\dagger = 1 - p_\star$, use $C = 10$, and vary the competitor dispersion

$$\xi = 1 - \|\boldsymbol{\rho}\|_2^2, \qquad 0 \le \xi \le 1 - \frac{1}{C-1}.$$

The retained, dropped, and full output-space traces are

$$\tau_{\mathrm{ret}} = p_\star p_\dagger (2 - \xi), \qquad \tau_{\mathrm{drop}} = p_\dagger \xi, \qquad \tau_{\mathrm{full}} = 2p_\star p_\dagger + p_\dagger^2 \xi.$$

The shaded gap in Figure 2(b) is $\tau_{\mathrm{full}} - \tau_{\mathrm{ret}} = \tau_{\mathrm{drop}}$.

**Panel (c): agreement between damped steps.** To isolate the effect of the dropped within-competitor covariance, we construct a two-dimensional example in which the gradient and FGN curvature remain fixed as competitor dispersion varies.

We use $C = 10$ and $p_\star = 0.70$. Let $\boldsymbol{A} \in \mathbb{R}^{C \times 2}$ be the logit Jacobian, with row $\boldsymbol{a}_c^\top$ for class $c$. We set

$$\boldsymbol{a}_\star = (-1, \eta)^\top, \qquad \boldsymbol{a}_j = (1, \beta t_j)^\top, \qquad \eta = 1.25, \qquad \beta = 3.0,$$

where $j = 1, \dots, 9$ indexes the competing classes and

$$t_1 = 0, \qquad \{t_2, \dots, t_9\} = \left\{\pm\tfrac{1}{4}, \pm\tfrac{1}{2}, \pm\tfrac{3}{4}, \pm 1\right\}.$$

We vary the competitor distribution using

$$\boldsymbol{\rho}(\alpha) = (1 - \alpha)\mathbf{e}_1 + \alpha \frac{\mathbf{1}_{C-1}}{C-1},$$

where $\mathbf{e}_1 \in \mathbb{R}^{C-1}$ is the first standard basis vector. We use 201 equally spaced values of $\alpha$ from $10^{-5}$ to 1, and plot against $\xi(\alpha) = 1 - \|\boldsymbol{\rho}(\alpha)\|_2^2$.

In the coordinate order $(\star, -\star)$, define

$$\boldsymbol{p} = \begin{bmatrix} p_\star \\ p_\dagger \boldsymbol{\rho} \end{bmatrix}, \qquad p_\dagger = 1 - p_\star, \qquad \boldsymbol{e}_\star = (1, 0, \dots, 0)^\top, \qquad \boldsymbol{g} = \boldsymbol{A}^\top (\boldsymbol{p} - \boldsymbol{e}_\star).$$

The two curvature matrices are

$$\boldsymbol{H}^{\mathrm{GGN}} = \boldsymbol{A}^\top \big(\mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top\big) \boldsymbol{A}$$

and

$$\boldsymbol{H}^{\mathrm{FGN}} = p_\star p_\dagger (\bar{\boldsymbol{a}}_\rho - \boldsymbol{a}_\star)(\bar{\boldsymbol{a}}_\rho - \boldsymbol{a}_\star)^\top, \qquad \bar{\boldsymbol{a}}_\rho = \boldsymbol{A}_{-\star}^\top \boldsymbol{\rho},$$

where $\boldsymbol{A}_{-\star}$ contains the competitor rows. Since $t_1 = 0$ and $\sum_j t_j = 0$, the mean competitor row $\bar{\boldsymbol{a}}_\rho = (1, 0)^\top$ is constant along the sweep.

For damping scales $s_\lambda \in \{0.01, 0.1, 1.0\}$, we set

$$\lambda_s = s_\lambda \frac{\mathrm{Tr}(\boldsymbol{H}^{\mathrm{FGN}}(\xi = 0))}{2},$$

using the limiting value at $\alpha \downarrow 0$, and hold $\lambda_s$ fixed along each curve. We compute both damped directions by direct $2 \times 2$ solves:

$$\boldsymbol{d}_{\mathrm{GGN}} = -(\boldsymbol{H}^{\mathrm{GGN}} + \lambda_s \boldsymbol{I}_2)^{-1} \boldsymbol{g}, \qquad \boldsymbol{d}_{\mathrm{FGN}} = -(\boldsymbol{H}^{\mathrm{FGN}} + \lambda_s \boldsymbol{I}_2)^{-1} \boldsymbol{g}.$$

Both directions are evaluated under the same full GGN damped quadratic model, with predicted decrease

$$\Delta_{\lambda_s}^{\mathrm{GGN}}(\boldsymbol{d}) \coloneqq -\left[\boldsymbol{g}^\top \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}^\top \boldsymbol{H}^{\mathrm{GGN}} \boldsymbol{d} + \frac{\lambda_s}{2} \|\boldsymbol{d}\|^2\right].$$

Panel (c) plots

$$R_{\lambda_s}(\xi) = \frac{\Delta_{\lambda_s}^{\mathrm{GGN}}(\boldsymbol{d}_{\mathrm{FGN}})}{\Delta_{\lambda_s}^{\mathrm{GGN}}(\boldsymbol{d}_{\mathrm{GGN}})}.$$

A ratio of one means that the two damped directions agree.

### F.2 Frozen-feature multiclass head optimization

**Data and features.** We use the canonical Cars196 [Krause et al., 2013] train/test split. Each image is passed once through a frozen ImageNet-pretrained ResNet-50 [He et al., 2016] encoder, using the corresponding TorchVision weight transform and no data augmentation. The cached feature is the average-pooled pre-classifier representation $\boldsymbol{h}_i \in \mathbb{R}^{2048}$. Features are standardized coordinatewise using the training-set mean and standard deviation. The cache contains $8144$ training features and $8041$ test features.

**Head objective.** Given feature $\boldsymbol{h}_i \in \mathbb{R}^{2048}$ and label $y_i \in \{1, \dots, 196\}$, the optimized head is

$$\boldsymbol{z}_i = \boldsymbol{W}^\top \boldsymbol{h}_i + \boldsymbol{b}, \qquad \boldsymbol{W} \in \mathbb{R}^{2048\times 196}, \qquad \boldsymbol{b} \in \mathbb{R}^{196}.$$

Equivalently, for row-wise batches,

$$\boldsymbol{Z} = \boldsymbol{H}\boldsymbol{W} + \mathbf{1}\boldsymbol{b}^\top.$$

The number of trainable parameters is

$$2048 \cdot 196 + 196 = 401604.$$

We minimize the mean multiclass softmax cross-entropy using mini-batches.

**Methods.** Adam [Kingma and Ba, 2014] provides a first-order baseline. SGN [Gargiani et al., 2020] uses the full softmax GGN with truncated parameter-space CG. FGN uses the retained true-vs-rest curvature from Theorem 3.1. For this affine head, the FGN row-space Gram has a closed form. Let $(\boldsymbol{a}_i, \boldsymbol{a}_j)$ denote the logit-space margin gradients for examples $(i, j)$, expressed in the same class ordering. The unwhitened Gram is

$$G_{ij} = (\boldsymbol{h}_i^\top \boldsymbol{h}_j + 1)(\boldsymbol{a}_i^\top \boldsymbol{a}_j),$$

where $+1$ accounts for the bias. The whitened row Gram is

$$K_{ij} = \sqrt{q_i q_j}\, G_{ij}, \qquad q_i = p_\star^{(i)} p_\dagger^{(i)}.$$

This closed-form expression is specific to the affine-head setting and introduces no additional curvature approximation.

**Hyperparameters.** Hyperparameters are selected on a train-only validation split and fixed for final test evaluation. The selected configurations are

| | | |
|---|---|---|
| Adam: | $\alpha = 3 \cdot 10^{-4}$, | |
| SGN: | $\alpha = 0.1,\ \lambda_0 = 1.0,$ | constant damping, CG maxiter $5$, |
| FGN: | $\alpha = 0.1,\ \lambda_0 = 1.0,$ | constant damping, CG maxiter $5$. |

Both second-order methods use CG tolerance $10^{-5}$. SGN uses warm-started CG, FGN uses row-space CG without warm-starting.

**Final runs and timing.** We train each method for $150$ epochs ($9600$ steps) across $10$ seeds, with batch size $128$ and evaluation every $100$ steps. Each seed determines head initialization and a single shuffle of the training examples; batches remain fixed across epochs. The final batch wraps $48$ examples from the beginning of the shuffled training array, giving $64$ batches per epoch.

Test accuracy and cross-entropy are interpolated onto a $240$-point wall-time grid ending at $7.27$ s, the shortest final time among the $30$ runs. Curves show means and $\pm 1$ standard deviation over seeds. Threshold times are computed from the interpolated mean accuracy curve. Wall time includes synchronized head updates but excludes feature extraction, compilation, evaluation, plotting, and CSV writes.

**Optimizer steps.** FGN reaches $44\%$ mean test accuracy after approximately $2443$ steps, compared with $2533$ for SGN. Both finish near $44.5\%$ after $9600$ steps. The wall-time advantage therefore comes primarily from cheaper updates, not substantially faster progress per step.

Table 2: **Frozen-feature head optimization on Cars196.** Test accuracy and cross-entropy at the shared 7.27 s horizon, reported as mean $\pm 1$ standard deviation over 10 seeds. Threshold times are in seconds.

| Method | Acc. (%) | CE | Time to 42% | Time to 43% | Time to 44% |
|---|---|---|---|---|---|
| Adam | $42.28 \pm 0.13$ | $2.8952 \pm 0.0076$ | 1.146 | – | – |
| SGN | $44.33 \pm 0.10$ | $2.3634 \pm 0.0006$ | 1.133 | 1.968 | 4.441 |
| FGN | $44.43 \pm 0.07$ | $2.3579 \pm 0.0005$ | 0.833 | 1.414 | 3.449 |

Table 3: **Cars196 CG budget sensitivity.** Final test accuracy, mean CG iterations, and ratio of mean SGN/FGN optimizer step times over three seeds. Paired entries are FGN / SGN.

| CG cap | Test accuracy (%) | CG iterations | SGN/FGN time |
|---|---|---|---|
| 1 | 44.26/43.99 | 1.00/1.00 | 1.14 |
| 2 | 44.33/44.30 | 2.00/2.00 | 1.17 |
| 5 | 44.51/44.55 | 5.00/5.00 | 1.35 |
| 10 | 44.52/44.50 | 8.83/9.15 | 1.47 |
| 20 | 44.51/44.48 | 11.40/12.74 | 1.62 |

**CG budget sensitivity.** We repeat the 9600-step runs over three seeds with CG caps $\{1, 2, 5, 10, 20\}$, keeping all other settings fixed.

At a CG cap of five, both methods exhaust the iteration budget in all solves. SGN performs one additional operator application per solve to accommodate warm starting, so equal CG caps do not imply equal computational work. Beyond five iterations, increasing the CG budget brings no consistent improvement in final test accuracy.

**Competitor dispersion.** We record competitor dispersion $\xi$, the effective competitor count $N_{\text{eff}} = 1/(1-\xi)$, and the dropped logit-space trace $p_\dagger \xi$ over all 8144 training examples, at initialization, every 500 steps, and at the final iterate, across 10 seeds. The effective competitor count is computed per example before averaging.

At initialization, mean $\xi \approx 0.995$ and $N_{\text{eff}} = 194.98$. At the final iterate, mean $N_{\text{eff}}$ is 10.33 for SGN and 10.94 for FGN. The largest competitor accounts for approximately 30% and 28% of the conditional competitor mass, respectively. For FGN, the dropped term still accounts for 40.85% of the full parameter-space GGN trace.

Therefore, in this setting, matching SGN's test accuracy does not require a nearly one-hot competitor distribution or a negligible discarded curvature term.

### F.3 End-to-end training

**Data and model.** We train a CIFAR-style ResNet-20 with 278,324 parameters on CIFAR-100 (50,000 training and 10,000 test images). Training augmentation includes padding, random cropping, and horizontal flips.

**Training protocol.** All methods train for 200 epochs with batch size 128, corresponding to 78,125 updates when epochs are counted by processed examples. The initial learning rate is 0.1 for SGD-M and 0.3 for SGN and FGN, with tenfold reductions at epochs 100 and 150. All methods use Nesterov momentum 0.9 and kernel-only weight decay $5 \times 10^{-4}$. Decay is applied before momentum for SGD-M and after momentum for SGN and FGN.

**Curvature solvers.** SGN applies the full softmax GGN with warm-started parameter-space CG. FGN uses the retained true-vs-rest curvature with the lifted row-space CG solver. Both methods use damping $\lambda = 1$, at most 5 CG iterations, and tolerance $10^{-5}$.

**Evaluation.** We use five seeds and evaluate training and test metrics at initialization and every 5 epochs. Test metrics cover all 10,000 test images. Training cross-entropy is measured on a fixed, unaugmented subset of 5000 training images, with BatchNorm in inference mode. We report the

Table 4: **ResNet-20 training on CIFAR-100.** Final metrics after 200 epochs and median training step times. Entries are means $\pm 1$ standard deviation over 5 seeds. Training CE is evaluated on the fixed training subset.

| Method | Test accuracy (%) | Test CE | Training CE | Step time (ms) |
|---|---|---|---|---|
| SGD-M | $68.23 \pm 0.34$ | $1.288 \pm 0.014$ | $0.2584 \pm 0.0054$ | $10.639 \pm 0.540$ |
| SGN | $66.78 \pm 0.23$ | $1.666 \pm 0.020$ | $0.2142 \pm 0.0050$ | $95.503 \pm 1.164$ |
| FGN | $67.05 \pm 0.35$ | $1.459 \pm 0.032$ | $0.2372 \pm 0.0042$ | $76.154 \pm 1.124$ |

final epoch rather than the best test checkpoint. Results are aggregated as mean $\pm$ sample standard deviation across seeds.

**Timing.** Step times include synchronized training and input overhead, but exclude compilation, evaluation, logging, and the first input wait of each timing segment. Within each run, we compute the mean step time per segment and take the median across segments. The resulting values are aggregated across seeds. Experiments run on an NVIDIA RTX A4000.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state the decomposition, PSD under-approximation, binary exactness, row-space solver, and empirical findings. These claims are supported by Sections 2–5 and Appendices B–F.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sections 1 and 3 discuss the approximation regime, while Section 5.1 examines how damping affects the resulting steps. Section 4.3 states the assumptions of the simplified convergence analysis. The empirical evaluation covers mechanism checks, a frozen-feature head benchmark, and end-to-end ResNet-20 training on CIFAR-100.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The main assumptions are stated in the theorem, proposition, and solver statements, including finite logits, $C \geq 2$, and $\lambda > 0$. Detailed proofs are given in Appendices B–D; the simplified convergence theorem states its assumptions explicitly.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix F describes the datasets, models, baselines, hyperparameters, solver settings, seeds, and timing conventions. Code to reproduce the experiments is available at `https://github.com/cor3bit/fgn`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The experiments use synthetic data and the existing Cars196 [Krause et al., 2013] and CIFAR-100 [Krizhevsky et al., 2009] datasets. Appendix F describes the experimental settings; code to reproduce the experiments is available at `https://github.com/cor3bit/fgn`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Appendix F specify the datasets, models, baselines, data splits, hyperparameters, solver settings, seeds, batching, and timing conventions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Figures 2(a) and 3(a,b) report means and standard deviations over 5, 10, and 5 seeds, respectively. Tables 2 and 4 provide the corresponding numerical results. The remaining mechanism plots are deterministic. Appendix F describes the aggregation and timing protocols.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F describes the compute setup (NVIDIA RTX A4000, 16 GB), timing protocol, and measured wall-clock update times.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and are not aware of any violations. The work introduces and evaluates an optimization method. It does not involve human-subject experiments or crowdsourcing, does not release high-risk models or scraped datasets, and does not introduce systems intended for sensitive deployment.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The work introduces an optimization method for multiclass cross-entropy. It does not introduce a deployed system, user-facing application, new dataset, pretrained model, or decision-making pipeline. Any societal effects are indirect and generic to improvements in machine learning training efficiency, rather than application-specific impacts introduced by this paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models, scraped datasets, or new data assets requiring controlled access. The experiments evaluate an optimization method using synthetic/mechanism studies and existing public image-classification benchmark data, so safeguards for responsible release of high-risk data or models are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper credits Cars196 [Krause et al., 2013], CIFAR-100 [Krizhevsky et al., 2009], and ResNet [He et al., 2016], and identifies TorchVision as the source of the pretrained weights. No third-party datasets or pretrained models are redistributed.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released assets are the FGN solver in Somax [Korbit and Zanon, 2026] and the experimental code linked in Section 5. The paper and accompanying repositories document their implementation and experimental settings.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or new research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects. No study participants are involved, so IRB or equivalent approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs. The proposed method, theoretical results, experimental protocol, data processing, baselines, and evaluation do not use LLMs as important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.