Jurnal Riset Mahasiswa Matematika
Volume 5 (2) (2025), Pages 102-122
p-ISSN: 2808-1552; e-ISSN: 2808-4926

Inexact Generalized Gauss–Newton–CG for Binary Cross-Entropy Minimization

Mohammad Jamhuri[1*], Silvi Puspita Sari[1], Siti Amiroch[2], Juhari[1], and Vivi Aida Fitria[3]

[1] *Department of Mathematics, Faculty of Science and Technology, UIN Maulana Malik Ibrahim, Malang, Indonesia*
[2] *Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Islam Darul Ulum, Lamongan, Indonesia*
[3] *Institut Teknologi dan Bisnis Asia, Malang, Indonesia*

Abstract

Binary cross-entropy (BCE) minimization is a standard objective in probabilistic binary classification, yet practical training pipelines often rely on first-order methods whose performance can be sensitive to step-size choices and may require many iterations to reach low-loss solutions. This paper studies an inexact curvature-based solver that combines a (generalized) Gauss–Newton approximation with conjugate gradient (CG) inner iterations for minimizing the regularized BCE objective in full-batch logistic regression. At each outer iteration, the method computes a descent direction by approximately solving a damped Gauss–Newton system in a matrix-free manner via repeated products with X and $X^\top$, and terminates CG according to a relative-residual inexactness rule. Numerical experiments on three benchmark datasets show that the proposed Inexact GGN–CG can substantially reduce the number of outer iterations on smaller numerical data, while remaining competitive in predictive performance, and can improve both validation and test mean BCE on larger mixed-type data after one-hot encoding. In particular, on Adult Census Income the method achieves lower test mean BCE (0.3176 ± 0.0044) and higher F1-score (0.6623 ± 0.0066) than Adam and gradient descent under the same regularization-selection protocol, at the cost of additional CG work. These results highlight how damping and inexactness jointly govern the trade-off between curvature-solve effort, wall-clock time, and achieved BCE values in deterministic logistic-regression training.

Keywords: generalized Gauss–Newton; conjugate gradient; inexact methods; binary cross-entropy; logistic regression; second-order optimization

1 Introduction

Binary classification problems are frequently formulated through probabilistic models that output an estimated class probability and are trained by minimizing the binary cross-entropy (BCE) objective [1], [2]. In the case of logistic regression, BCE coincides with the negative log-likelihood of a Bernoulli model with a sigmoid link, providing a statistically principled loss that is well aligned with probability calibration [1]. Despite the convexity of BCE for logistic regression, obtaining high-accuracy solutions efficiently can still be challenging in practice when feature dimension is large, data are ill-conditioned, or when strict tolerances on the objective or gradient norm are required for downstream tasks [3].

A common approach to BCE minimization is to use first-order optimization methods such as gradient descent variants and adaptive stochastic optimizers [4]. These methods are attractive

*Corresponding author. E-mail: m.jamhuri@live.com

Submitted: June 23, 2025 Reviewed: November 21, 2025 Accepted: December 21, 2025
DOI: https://dx.doi.org/10.18860/jrmm.v5i2.34739

due to their low per-iteration cost and minimal linear algebra requirements, but they can exhibit slow convergence near the optimum and can be sensitive to hyperparameter choices such as learning rates, batch sizes, and decay schedules. By contrast, second-order methods exploit curvature information to produce search directions that more directly reflect the local geometry of the objective, often leading to faster local convergence and improved robustness to scaling in the parameter space. The primary obstacle is that second-order updates typically require solving linear systems involving Hessian-related matrices, which can be computationally expensive or memory-intensive if treated with direct factorizations, especially as the parameter dimension grows [5], [6].

To address this bottleneck, this paper considers a curvature approximation and an inexact linear solve strategy that together yield an efficient and stable algorithm for BCE minimization. Specifically, we employ the generalized Gauss–Newton (GGN) approximation, which can be viewed as a structured positive semidefinite surrogate for the true Hessian that is particularly natural for objectives expressed as a composition of a prediction map with a convex loss [7], [8], [9], [10], [11]. The resulting linear system is not solved exactly; instead, we compute an approximate step using the conjugate gradient (CG) method and terminate the inner iterations according to an inexactness criterion.

Recent work has also explored scalable second-order or curvature-aware optimizers and Gauss–Newton-style updates in modern machine-learning pipelines [12], [13], [14]. These approaches are motivated by the observation that first-order methods can be sensitive to step-size choices and may converge slowly in ill-conditioned regimes, while curvature information can provide better-scaled updates [15]. At the same time, they avoid the classical bottleneck of second-order methods by relying on matrix-free curvature–vector products and approximate inner solves, rather than explicit Hessian formation or direct factorizations [16]. This perspective aligns with inexact GGN strategies: early termination of inner iterations can reduce computational cost substantially, provided that the resulting direction remains a descent direction and is globalized via line search or damping. In this paper, we investigate this trade-off in a deterministic and convex setting (regularized logistic regression), allowing the roles of damping and inexactness to be examined with minimal confounding factors.

The contributions of this work are threefold. First, we formulate an inexact GGN–CG method tailored to BCE minimization and specialize it to logistic regression, providing a baseline setting where the underlying operators and convergence behavior can be studied clearly. Second, we present a matrix-free implementation in which the action of the damped GGN operator on a vector is computed via inexpensive matrix–vector products, enabling the use of CG without explicit formation of the curvature matrix. Third, we conduct a computational study that compares the proposed method to standard first-order baselines under controlled preprocessing and stopping criteria, with emphasis on the interplay between damping, inexactness tolerance, runtime, and the achieved objective value.

The remainder of the paper is organized as follows. The next section introduces the BCE objective for logistic regression and reviews the GGN approximation in this context. The methodology section then details the inexact GGN–CG algorithm, including damping, the CG stopping rule, and practical considerations for stable implementation. The experimental section describes datasets, preprocessing, baseline methods, and evaluation metrics, followed by numerical results and discussion. The paper concludes with a summary of findings and directions for subsequent work.

2 Problem Formulation and Background

This work considers binary classification with inputs $x_i \in \mathbb{R}^n$ and labels $y_i \in \{0, 1\}$ for $i = 1, \ldots, m$. Let $X \in \mathbb{R}^{m \times n}$ denote the design matrix whose ith row is $x_i^\top$, and let $y \in \mathbb{R}^m$ collect the labels. The baseline model is logistic regression, in which the logit is linear in the parameters $\theta \in \mathbb{R}^n$

and defined by

$$z_i(\theta) = x_i^\top \theta, \qquad z(\theta) = X\theta. \tag{1}$$

The predicted probability of the positive class is given by the sigmoid mapping $p_i(\theta) = \sigma(z_i(\theta))$, where $\sigma(t) = (1 + e^{-t})^{-1}$. Writing $p(\theta) \in \mathbb{R}^m$ for the vector of probabilities, we have $p(\theta) = \sigma(X\theta)$ with $\sigma(\cdot)$ applied componentwise.

2.1 Binary cross-entropy objective

The empirical binary cross-entropy (BCE) objective for logistic regression is

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell_i(\theta), \qquad \ell_i(\theta) = -\Big(y_i \log p_i(\theta) + (1 - y_i) \log\big(1 - p_i(\theta)\big)\Big). \tag{2}$$

To improve numerical stability and to control model complexity, we optionally consider ℓ_2-regularization, leading to

$$L_\beta(\theta) = L(\theta) + \frac{\beta}{2}\|\theta\|_2^2, \tag{3}$$

where $\beta \geq 0$ is a regularization parameter. For logistic regression, $L(\theta)$ is convex in θ, and $L_\beta(\theta)$ is strongly convex when $\beta > 0$, which is advantageous for conditioning and uniqueness of the minimizer [1], [3].

The gradient of the BCE objective admits a compact form. Using the identity $\nabla_\theta z(\theta) = X$, one obtains

$$\nabla L(\theta) = \frac{1}{m} X^\top (p(\theta) - y), \tag{4}$$

and, with ℓ_2-regularization,

$$g(\theta) \;:=\; \nabla L_\beta(\theta) = \frac{1}{m} X^\top (p(\theta) - y) + \beta\theta. \tag{5}$$

These expressions highlight that gradient evaluation requires only matrix–vector products with X and $X^\top$, together with the componentwise sigmoid computation [17].

2.2 Curvature structure and the generalized Gauss–Newton approximation

Second-order methods for minimizing $L_\beta(\theta)$ exploit curvature information through the Hessian (or a structured approximation). For logistic regression with mean BCE, the curvature admits a convenient weighted form involving the design matrix and a diagonal weight matrix. For completeness, we outline how this structure arises.

We start from the mean BCE written in the numerically convenient form

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m} \Big(\log\big(1 + \exp(z_i)\big) - y_i z_i\Big), \qquad z_i = x_i^\top \theta, \tag{6}$$

so that $p_i = \sigma(z_i)$ and $\nabla_\theta z_i = x_i$. Define $\phi_i(z_i) := \log(1 + \exp(z_i)) - y_i z_i$. Then

$$\frac{d\phi_i}{dz_i} = p_i - y_i, \qquad \frac{d^2\phi_i}{dz_i^2} = p_i(1 - p_i) =: w_i, \tag{7}$$

with $0 < w_i \leq \frac{1}{4}$. Recalling that $\nabla L(\theta) = \frac{1}{m} X^\top (p(\theta) - y)$, differentiating once more yields

$$\nabla^2 L(\theta) = \frac{1}{m}\sum_{i=1}^{m} w_i\, x_i x_i^\top = \frac{1}{m} X^\top W(\theta) X, \qquad W(\theta) = \operatorname{diag}(w_1, \ldots, w_m). \tag{8}$$

With ℓ_2-regularization $L_\beta(\theta) = L(\theta) + \frac{\beta}{2}\|\theta\|_2^2$, the penalty contributes $\nabla^2\Big(\frac{\beta}{2}\|\theta\|_2^2\Big) = \beta I$, hence

$$H_\beta(\theta) = \nabla^2 L_\beta(\theta) = \frac{1}{m} X^\top W(\theta) X + \beta I. \tag{9}$$

Generalized Gauss–Newton (GGN). The generalized Gauss–Newton approximation provides a positive semidefinite curvature model for objectives expressed as a convex loss applied to model outputs [7], [8], [9], [10]. In logistic regression, the model map is the linear logit $z(\theta) = X\theta$, and the scalar BCE loss is convex in each z_i. In this case, the GGN matrix coincides with the exact Hessian of $L(\theta)$, and the regularized GGN is therefore

$$G_\beta(\theta) = \frac{1}{m} X^\top W(\theta) X + \beta I \;=\; H_\beta(\theta). \tag{10}$$

Implication for scalable solvers. The representation $X^\top W(\theta)X$ is attractive computationally because it enables matrix-free curvature–vector products via multiplications with X and $X^\top$, together with inexpensive elementwise weighting by $w(\theta)$. In the next subsection, we use this structure to define a damped curvature system and show how its action on a vector can be evaluated without explicitly forming $X^\top W(\theta)X$.

2.3 Damping and the linear system underlying GGN steps

Building on the curvature structure above, at iterate θ_k we compute a curvature-informed direction by solving a damped GGN system. In general the damping can be iteration-dependent; in our experiments we use a fixed value (i.e., $\lambda_k \equiv \lambda$ within each run).

$$(G_\beta(\theta_k) + \lambda_k I)s_k = -g(\theta_k), \tag{11}$$

where $\lambda_k > 0$ is a damping parameter. The term $\lambda_k I$ ensures strict positive definiteness of the system matrix and improves conditioning, thereby supporting robust iterative solution by conjugate gradient (CG). Importantly, the left-hand side of (11) need not be formed explicitly. For any vector $v \in \mathbb{R}^n$, the matrix–vector product required by CG is

$$(G_\beta(\theta_k) + \lambda_k I)v \;=\; \frac{1}{m} X^\top (W(\theta_k)(Xv)) + (\beta + \lambda_k)v. \tag{12}$$

which depends only on matrix–vector products with X and $X^\top$ and componentwise multiplication by the diagonal weights. This matrix-free structure is central to the scalability of the proposed inexact GGN–CG approach, and it enables the use of iterative linear algebra without storing or factorizing curvature matrices [5], [6].

Equations (11)–(12) show that the curvature step for regularized logistic regression can be computed through matrix–vector products with X and $X^\top$ together with elementwise weighting by $W(\theta)$. This matrix-free representation motivates an iterative linear solver in the inner loop, avoiding explicit formation or factorization of curvature matrices.

In the next section, we instantiate this idea into a practical inexact second-order method by solving the damped generalized Gauss–Newton system approximately using conjugate gradient (CG) and globalizing the update with Armijo backtracking line search.

3 Proposed Method: Inexact GGN–CG

This section describes an inexact generalized Gauss–Newton method in which the curvature system is solved approximately by conjugate gradient (CG). We focus on regularized logistic regression with a *mean* binary cross-entropy objective. In the numerical study, the damping level and CG forcing tolerance are selected by validation and then held *constant* throughout the outer iterations of each run.

Although the formulation in Section 2 admits iteration-dependent damping (as in (11)), in our numerical study we select a single damping level λ by validation and keep it fixed within each run for interpretability and controlled comparison across solvers.

Figure 1 provides an overview of the outer–inner structure. It highlights the matrix-free application of the damped GGN operator within CG, the relative-residual stopping rule for the inner solve, and globalization via Armijo backtracking line search with a descent safeguard.

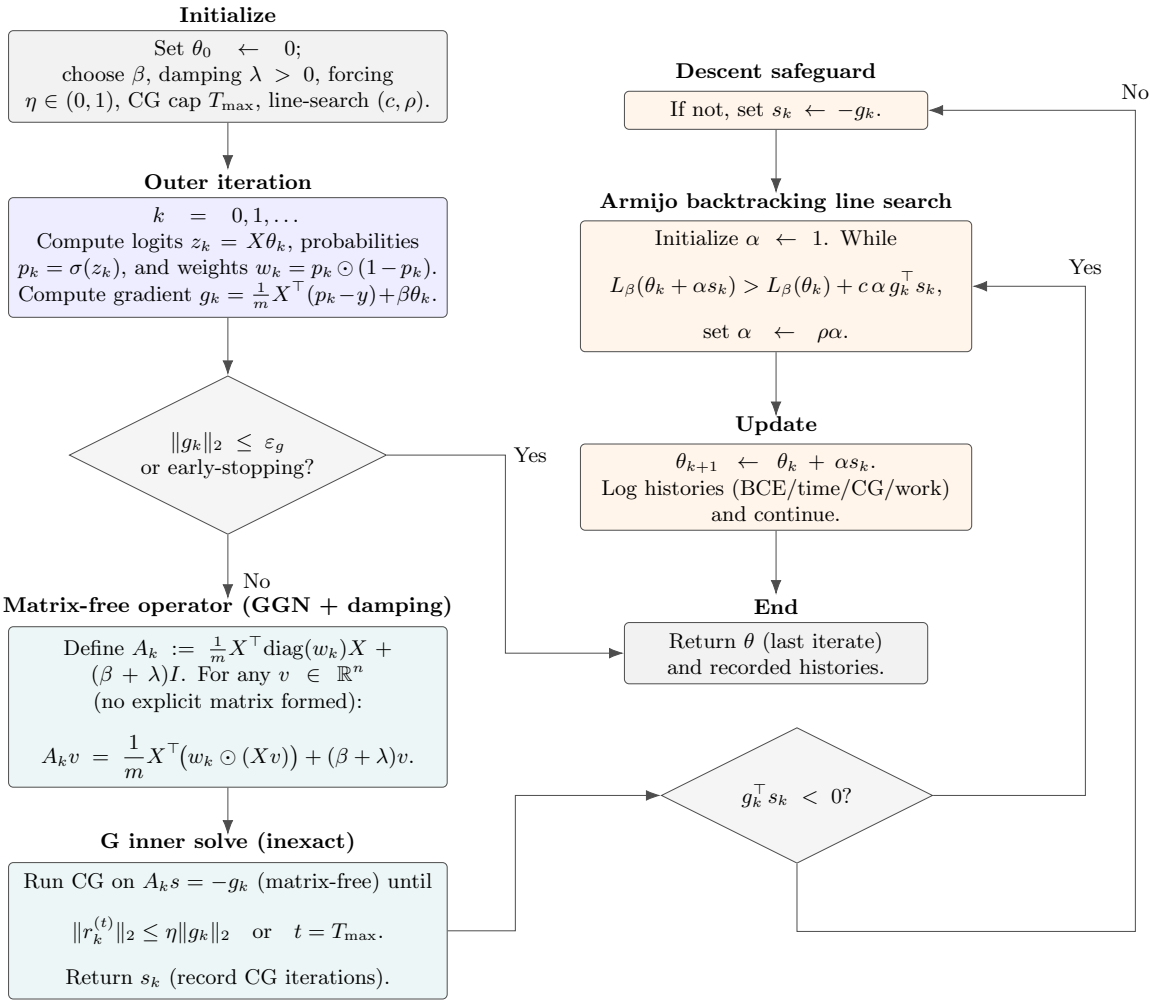


Figure 1: Flowchart of the proposed Inexact GGN–CG method for regularized logistic regression. Each outer iteration forms w_k and g_k, applies the damped GGN operator A_k in a matrix-free manner within CG, terminates the inner solve using the relative-residual rule $\|r_k^{(t)}\|_2 \le \eta \|g_k\|_2$, and globalizes the step using Armijo backtracking with a descent safeguard.

3.1 Damped generalized Gauss–Newton step

Let θ_k be the current iterate and define the regularized training objective $L_\beta(\theta) = L(\theta) + \frac{\beta}{2}\|\theta\|_2^2$, where $L(\theta)$ is the mean BCE. The method computes a search direction s_k by approximately solving the damped generalized Gauss–Newton system

$$A_k s_k = -g_k, \qquad A_k := G_\beta(\theta_k) + \lambda I, \qquad g_k := \nabla L_\beta(\theta_k), \tag{13}$$

where $\lambda > 0$ is a (run-wise) damping parameter. For logistic regression,

$$G_\beta(\theta_k) = \frac{1}{m} X^\top W_k X + \beta I, \qquad W_k = \mathrm{diag}(w_k), \qquad w_k = p_k \odot (1 - p_k), \quad p_k = \sigma(X\theta_k). \tag{14}$$

Equivalently,

$$A_k = \frac{1}{m} X^\top W_k X + (\beta + \lambda) I. \tag{15}$$

Note that A_k changes with k through W_k, while λ is fixed within each run. The damping term λI ensures that the system matrix is strictly positive definite and improves conditioning; see Lemma 1.

For CG, we only require matrix–vector products. For any $v \in \mathbb{R}^n$,

$$A_k v = \frac{1}{m} X^\top (W_k (Xv)) + (\beta + \lambda) v. \tag{16}$$

Since $W_k = \operatorname{diag}(w_k)$, we implement $W_k(Xv)$ as the elementwise product $w_k \odot (Xv)$.

3.2 Conjugate gradient and the inexactness criterion

We apply CG to (13) in a matrix-free manner. By Lemma 1, A_k is symmetric positive definite for all k when $\lambda > 0$, hence CG is well-defined [5]. Let $s_k^{(t)}$ denote the t-th CG iterate and

$$r_k^{(t)} := A_k s_k^{(t)} + g_k \tag{17}$$

the linear-system residual. CG is terminated early using the relative residual rule

$$\|r_k^{(t)}\|_2 \le \eta \|g_k\|_2, \tag{18}$$

where $\eta \in (0,1)$ is a forcing parameter (held constant within a run), together with a hard cap $t \le T_{\max}$ [5], [18]. Smaller η typically increases inner iterations but can reduce outer iterations, while larger η reduces inner cost at the risk of slower outer progress. In all experiments, η is selected by validation and then fixed within a run.

3.3 Step acceptance via backtracking line search

Given an approximate direction s_k, we update

$$\theta_{k+1} = \theta_k + \alpha_k s_k, \tag{19}$$

where α_k is chosen by Armijo backtracking line search on the *training* objective L_β [5]:

$$L_\beta(\theta_k + \alpha_k s_k) \le L_\beta(\theta_k) + c\, \alpha_k\, g_k^\top s_k, \tag{20}$$

with $c \in (0,1)$ and reduction factor $\rho \in (0,1)$. If numerical issues or inexact solves produce a non-descent direction ($g_k^\top s_k \ge 0$), we enforce a descent safeguard by setting $s_k \leftarrow -g_k$, yielding $g_k^\top s_k = -\|g_k\|_2^2 < 0$. Proposition 2 guarantees that, for any descent direction, backtracking accepts a step size in finitely many reductions and ensures strict decrease in L_β.

3.4 Algorithm summary and basic guarantees

Algorithm 1 summarizes the full method used in the numerical study. The training objective includes ℓ_2-regularization, while validation and test losses are reported as mean BCE *without* the penalty term. All computations are full-batch and use a matrix-free representation of the generalized Gauss–Newton operator via products with X and $X^\top$.

Proposition 1 (Curvature structure for mean BCE)**.** *Let $L(\theta)$ denote the* mean *BCE for logistic regression,*

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \Big(\log\left(1 + \exp(x_i^\top \theta)\right) - y_i\, x_i^\top \theta \Big), \qquad p(\theta) = \sigma(X\theta).$$

Then

$$\nabla L(\theta) = \frac{1}{m} X^\top (p(\theta) - y), \qquad \nabla^2 L(\theta) = \frac{1}{m} X^\top W(\theta) X,$$

where $W(\theta) = \operatorname{diag}(p_i(\theta)(1 - p_i(\theta))) \succeq 0$ and $0 < p_i(\theta)(1 - p_i(\theta)) \le \frac{1}{4}$ for all i and all θ. Moreover, for this model and loss, the generalized Gauss–Newton matrix coincides with $\nabla^2 L(\theta)$.

Algorithm 1 Inexact GGN–CG with Armijo line search for regularized logistic regression

Input: $X \in \mathbb{R}^{m\times n}$ (including intercept), $y \in \{0,1\}^m$, regularization $\beta \geq 0$, damping $\lambda > 0$, forcing parameter $\eta \in (0,1)$, CG cap $T_{\max}$, outer budget $K_{\max}$, tolerance $\varepsilon_g > 0$, Armijo $c \in (0,1)$, backtracking $\rho \in (0,1)$

Output: parameters θ

$\theta_0 \leftarrow 0 \in \mathbb{R}^n$
for $k = 0, 1, \ldots, K_{\max} - 1$ **do**
  $z_k \leftarrow X\theta_k$, $p_k \leftarrow \sigma(z_k)$, $w_k \leftarrow p_k \odot (1 - p_k)$
  $g_k \leftarrow \frac{1}{m} X^\top (p_k - y) + \beta\theta_k$
  if $\|g_k\|_2 \leq \varepsilon_g$ **then**
    break
  end if
  Define $A_k v := \frac{1}{m} X^\top (w_k \odot (Xv)) + (\beta + \lambda)v$
  Use CG to approximately solve $A_k s_k = -g_k$ with $\|r_k^{(t)}\|_2 \leq \eta \|g_k\|_2$ and $t \leq T_{\max}$
  if $g_k^\top s_k \geq 0$ **then**
    $s_k \leftarrow -g_k$ ▷ descent safeguard
  end if
  $\alpha \leftarrow 1$
  while $L_\beta(\theta_k + \alpha s_k) > L_\beta(\theta_k) + c\,\alpha\, g_k^\top s_k$ **do**
    $\alpha \leftarrow \rho\,\alpha$
  end while
  $\theta_{k+1} \leftarrow \theta_k + \alpha s_k$
end for
return θ_k ▷ last available iterate (after break or after $K_{\max}$ steps)

Proof. Write $z = X\theta$ and use $L(\theta) = \frac{1}{m}\sum_{i=1}^m \big(\log(1 + \exp(z_i)) - y_i z_i\big)$. Differentiating gives $\nabla L(\theta) = \frac{1}{m} X^\top(\sigma(z) - y)$. Differentiating again yields $\nabla^2 L(\theta) = \frac{1}{m} X^\top W(\theta) X$ with $W(\theta) = \operatorname{diag}(\sigma(z_i)(1 - \sigma(z_i)))$. Since $\sigma(t)(1 - \sigma(t)) \in (0, \frac{1}{4}]$ for all $t \in \mathbb{R}$, we have $W(\theta) \succeq 0$. Because the model map $z(\theta) = X\theta$ is linear and the scalar loss is convex in z_i, the generalized Gauss–Newton construction reduces to the same curvature term, hence it coincides with $\nabla^2 L(\theta)$ in this setting. □

Lemma 1 (SPD property under regularization and damping)**.** *Fix $\beta \geq 0$ and $\lambda > 0$, and define $A(\theta) = \frac{1}{m} X^\top W(\theta) X + (\beta + \lambda) I$. Then $A(\theta)$ is symmetric positive definite for every $\theta \in \mathbb{R}^n$.*

Proof. Symmetry is immediate. For any nonzero $v \in \mathbb{R}^n$,

$$v^\top A(\theta) v = \frac{1}{m}(Xv)^\top W(\theta)(Xv) + (\beta + \lambda)\|v\|_2^2.$$

The first term is nonnegative since $W(\theta) \succeq 0$, and the second term is strictly positive because $\beta + \lambda > 0$. Therefore $v^\top A(\theta) v > 0$ for all $v \neq 0$, hence $A(\theta) \succ 0$. □

Proposition 2 (Existence of an Armijo-accepted step)**.** *Assume $L_\beta(\theta) = L(\theta) + \frac{\beta}{2}\|\theta\|_2^2$ is continuously differentiable. If s_k satisfies $g_k^\top s_k < 0$, then Armijo backtracking line search with (20) accepts a step size $\alpha > 0$ in finitely many reductions. Consequently, $L_\beta(\theta_{k+1}) < L_\beta(\theta_k)$.*

Proof. Since L_β is continuously differentiable, the directional derivative at $\alpha = 0$ is $\frac{d}{d\alpha} L_\beta(\theta_k + \alpha s_k)\big|_{\alpha=0} = g_k^\top s_k < 0$. Thus there exists $\bar\alpha > 0$ such that for all $\alpha \in (0, \bar\alpha]$ the Armijo inequality holds. Backtracking generates a geometric sequence $\alpha, \rho\alpha, \rho^2\alpha, \ldots$, which must enter $(0, \bar\alpha]$ in finitely many steps, hence it accepts some $\alpha > 0$. Moreover, the right-hand side of Armijo is strictly less than $L_\beta(\theta_k)$ when $\alpha > 0$ and $g_k^\top s_k < 0$, so the accepted step yields strict decrease. □

Table 1: Summary of datasets used in the experimental study. The feature dimension n is reported after preprocessing and includes an intercept term. The positive-class proportion is $\pi = \frac{1}{m}\sum_{i=1}^{m} y_i$.

Dataset	m	n	π
Pima Indians Diabetes	768	9	0.3490
Breast Cancer Wisconsin	569	31	0.3726
Adult Census Income	32 561	108	0.2408

3.5 Stopping conditions and practical parameterization

The outer loop terminates when

$$\|g_k\|_2 \le \varepsilon_g, \tag{21}$$

or when the outer-iteration budget $K_{\max}$ is reached. In the experiments, we additionally employ a stagnation-based early-stopping rule on the *reported* training mean BCE: the solver terminates if the relative improvement stays below a fixed tolerance for a fixed number of consecutive iterations. The stagnation test is applied to the reported mean BCE (without the ℓ_2 term), while the line search enforces descent on L_β. The damping λ, forcing parameter η, and CG cap $T_{\max}$ are treated as hyperparameters selected on the validation set and then fixed within each run.

3.6 Computational cost per iteration

The dominant cost in each outer iteration is the sequence of CG matrix–vector products with A_k. Each CG iteration evaluates (16), requiring one multiplication by X and one multiplication by $X^\top$, plus elementwise operations for the weights. Consequently, runtime is governed primarily by the total number of Xv and $X^\top u$ products, motivating our reporting of both wall-clock time and the accumulated CG iterations.

4 Experimental Setup

This section describes the experimental protocol used to evaluate the proposed Inexact GGN–CG method for binary cross-entropy (BCE) minimization in regularized logistic regression. All solvers are implemented within a single codebase to ensure consistent preprocessing, objective evaluation, and timing procedures. Unless stated otherwise, all reported results are aggregated over multiple random seeds and presented as mean ± standard deviation.

4.1 Datasets

Experiments are conducted on three benchmark binary classification datasets that exhibit distinct scales and feature types: *Pima Indians Diabetes*, *Breast Cancer Wisconsin*, and *Adult Census Income*. Pima Indians Diabetes is a small-to-medium, fully numerical tabular dataset. Breast Cancer Wisconsin is a small numerical dataset with strong signal-to-noise and near-separability in several splits. Adult Census Income is substantially larger and contains both numerical and categorical features, leading to a higher-dimensional representation after one-hot encoding.

For label conventions, we use the dataset-provided binary outcome for Pima Indians Diabetes. For Breast Cancer Wisconsin, we map malignant (“M”) to 1 and benign (“B”) to 0. For Adult Census Income, we map income $>$ 50K to 1 and $\le$ 50K to 0 after stripping whitespace and trailing punctuation in the label field. Table 1 reports the total number of samples m, the number of features n after preprocessing *including the intercept term*, and the positive-class prevalence π.

4.2 Data splitting and preprocessing

For each dataset, we create stratified splits into training, validation, and test sets with proportions 60%/20%/20%. Concretely, we first split the full dataset into a 60% training set and a 40%

temporary set, and then split the temporary set evenly into validation and test sets. Results are reported over five random seeds (seeds 0–4), and all summary statistics are computed across these seeds.

All preprocessing operations are fitted using the training split only and then applied unchanged to the validation and test splits to avoid information leakage. Numerical features are standardized using the training-set mean and standard deviation. For the Adult dataset, categorical features are one-hot encoded; categories not observed in the training split are ignored at transform time. Missing categorical entries represented by the symbol "?" are treated as an explicit "Unknown" category, and missing numerical values are imputed using the training-set median.

After preprocessing, an intercept is incorporated by augmenting the design matrix with a column of ones, so that the parameter vector includes a bias term. Consistent with the implementation used to generate the reported results, the ℓ_2 penalty is applied to *all* parameters, including the intercept.

4.3 Solvers

We compare three deterministic full-batch solvers for minimizing a regularized logistic-regression objective. The proposed method is the inexact generalized Gauss–Newton method with conjugate gradient inner solves and Armijo backtracking line search, denoted *Inexact GGN–CG*. Two first-order baselines are included: full-batch gradient descent (GD) and Adam.

All methods optimize the same *regularized training objective*

$$L_\beta(\theta) \;=\; L(\theta) + \frac{\beta}{2}\|\theta\|_2^2, \tag{22}$$

where $L(\theta)$ is the *mean* BCE on the training set. For GD, we use a stabilized step-size safeguard: if a tentative update increases the regularized training objective, the effective learning rate is repeatedly halved (up to a fixed number of reductions) until the objective decreases. For Adam, we use the standard moment parameters $(\beta_1, \beta_2, \epsilon) = (0.9, 0.999, 10^{-8})$.

For Inexact GGN–CG, the direction is computed by approximately solving the damped curvature system using CG with a relative residual stopping rule, and step acceptance is enforced by Armijo backtracking line search on $L_\beta(\theta)$ (with fixed Armijo parameters $c = 10^{-4}$ and backtracking factor $\rho = 0.5$, capped at a fixed maximum number of backtracking steps).

Curvature-aware optimization has also been studied in scalable second-order optimizers for machine learning, motivating our comparison between a Gauss–Newton–CG style method and first-order baselines [12], [13].

4.4 Regularization and hyperparameter selection

Regularization is controlled by an ℓ_2 penalty parameter $\beta \geq 0$ applied to all parameters. The key design choice in this study is that β is selected *once per dataset and seed* and then held fixed across all solvers for that seed, ensuring that methods are compared on the same regularized objective.

The selection procedure is as follows. For each candidate β in the grid

$$\beta \in \{0, 10^{-4}, 10^{-3}, 10^{-2}\},$$

we tune each solver on the training split and evaluate performance on the validation split using the validation mean BCE without the penalty term. We then define the validation score of β as the best (minimum) validation mean BCE achieved by any of the solvers under that β, and select β^* as the minimizer of this score. After choosing β^* for a given dataset and seed, we retain for each solver its own validation-selected hyperparameters under β^* and use these settings for test evaluation.

Given β^*, solver-specific hyperparameters are chosen by validation search over the following grids:

- `GD:` learning rate `lr` $\in \{10^{-4}, 3\cdot10^{-4}, 10^{-3}, 3\cdot10^{-3}, 10^{-2}, 3\cdot10^{-2}, 10^{-1}\}$.
- `Adam:` learning rate `lr` $\in \{10^{-4}, 3\cdot10^{-4}, 10^{-3}, 3\cdot10^{-3}, 10^{-2}\}$.
- `Inexact GGN-CG:` damping $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$, forcing tolerance $\eta \in \{10^{-1}, 10^{-2}, 10^{-3}\}$ in the relative residual rule $\|r\|_2 \le \eta\|g\|_2$, and CG iteration cap $T_{\max} \in \{50, 100, 200\}$.

Within each run of Inexact GGN–CG, the selected damping λ and forcing tolerance η are held constant throughout the outer iterations.

4.5 Stopping criteria and iteration budgets

All methods are run with dataset-specific iteration budgets to balance runtime and comparability across seeds. For Pima Indians Diabetes and Breast Cancer Wisconsin, the maximum number of outer iterations is set to 300, whereas for Adult Census Income it is set to 120 to keep runtimes practical given the larger sample size and one-hot encoded representation.

A stationarity tolerance is imposed via the gradient norm condition

$$\|\nabla L_\beta(\theta)\|_2 \le 10^{-4}. \tag{23}$$

In addition, an early-stopping criterion based on stagnation of the *reported* training mean BCE is used: if the relative improvement in training mean BCE (computed *without* the ℓ_2 term) falls below 10^{-10} for eight consecutive outer iterations, the solver terminates. For Inexact GGN–CG, the inner CG loop terminates when either the relative residual condition is met or the iteration cap $T_{\max}$ is reached; the accumulated number of CG iterations is recorded as an additional proxy for linear-algebra effort.

4.6 Evaluation metrics and reporting

Optimization efficiency is quantified by (i) the number of outer iterations, (ii) the total number of CG iterations accumulated across outer iterations for Inexact GGN–CG, and (iii) wall-clock training time measured consistently across methods. Time is measured around the solver loop and includes all computations performed by the solver (including line search and CG iterations) but excludes dataset loading and preprocessing.

Predictive performance is assessed on the held-out test set using the mean BCE *without* the ℓ_2 penalty term, and threshold-based classification metrics obtained by predicting class labels via $\hat{y} = \mathbb{I}(p \ge 0.5)$. We report accuracy, precision, recall, and F1-score. All reported metrics are aggregated over the five random seeds and presented as mean $\pm$ standard deviation. In addition, convergence plots report training mean BCE versus outer iteration and versus wall-clock time, averaged across seeds.

5 Results and Discussion

This section reports empirical results for binary cross-entropy minimization in logistic regression and analyzes the optimization behavior and predictive performance of *Inexact GGN–CG* relative to full-batch first-order baselines (GD and Adam). All results follow the protocol in Section 4, using stratified 60%/20%/20% splits, training-set standardization, validation-based hyperparameter selection, and reporting as mean $\pm$ standard deviation across five random seeds.

To avoid long sequences of floating objects, we organize results by dataset. Within each dataset, we first discuss optimization efficiency (outer iterations, wall-clock time, and CG work), then convergence trajectories, and finally held-out predictive performance. For readability, the two trajectory plots (loss versus iteration and loss versus time) are grouped as subfigures within a single figure per dataset. Note that reaching the outer-iteration cap does not necessarily imply lack of convergence; rather, it indicates that the common stopping rules (stationarity and

stagnation tolerances) were not triggered earlier under the selected hyperparameters and fixed budgets, which we keep consistent across solvers for comparability.

5.1 Cross-dataset trade-off and a hardware-agnostic work proxy

Wall-clock time is the most practical efficiency metric, but it can vary substantially across environments (CPU model, BLAS backend, sparse-matrix routines, and I/O). To complement time-based reporting, we therefore summarize efficiency using a simple, implementation-agnostic *work proxy* that captures the dominant linear-algebra cost shared by all solvers in this study: repeated products with the design matrix X and its transpose $X^\top$. This proxy is particularly relevant for Inexact GGN–CG, where each inner CG iteration requires one multiplication by X and one multiplication by $X^\top$ to evaluate the damped GGN operator in a matrix-free manner.

Concretely, we define the cumulative work as the approximate number of $(Xv + X^\top u)$ products incurred over the course of training. We assign a baseline cost of 2 such products per outer iteration (to reflect the main gradient-related products needed to compute logits and the gradient), and for Inexact GGN–CG we add 2 additional products per CG iteration. This yields the per-run work proxy

$$\text{work} \;\approx\; 2\,K \;+\; 2\sum_{k=1}^{K} t_k, \tag{24}$$

where K is the number of outer iterations and t_k is the number of CG iterations taken at outer iteration k (with $t_k = 0$ for GD and Adam). While this proxy does not account for every auxiliary objective evaluation (e.g., backtracking steps), it aligns with the dominant cost driver in all methods—sparse matrix–vector multiplications—and provides a consistent lens for comparing runs across datasets and solvers.

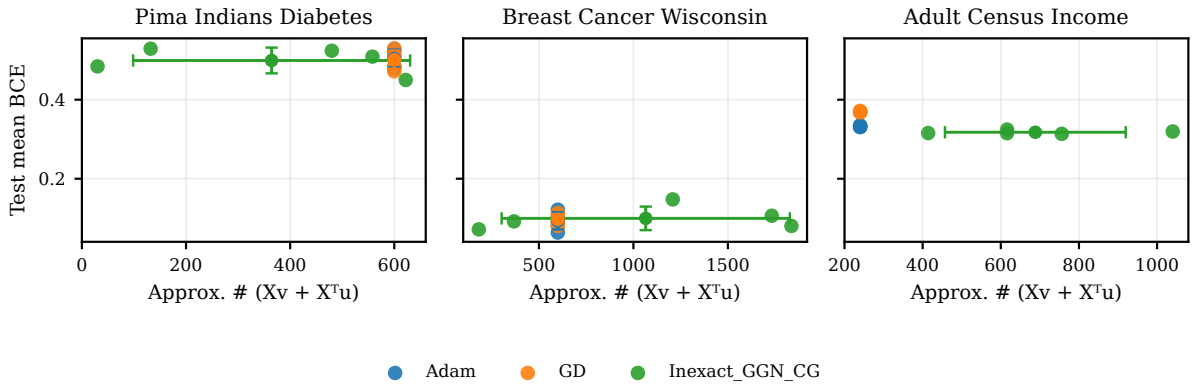


Figure 2: Accuracy–effort trade-off across datasets. Each point corresponds to one random seed. The horizontal axis reports the work proxy in (24) (approx. number of $(Xv + X^\top u)$ products), while the vertical axis reports test mean BCE. Error bars indicate mean ± one standard deviation across seeds.

Figure 2 summarizes the resulting effort–accuracy trade-off across datasets. On Pima Indians Diabetes, curvature-informed steps can reduce the number of outer iterations substantially, and the added CG effort is often offset by the decreased outer-loop budget, yielding competitive (sometimes lower) total work at comparable test BCE. On Breast Cancer Wisconsin, the problem is small and comparatively benign, so first-order baselines remain competitive and the additional CG work does not reliably translate into improved test BCE. On Adult Census Income, Inexact GGN–CG expends more work due to inner solves, but this additional linear-algebra effort is consistently associated with lower test mean BCE, matching the improvements observed in validation and downstream metrics reported in the dataset-specific subsections below.

5.2 Pima Indians Diabetes

Pima Indians Diabetes is a medium-scale, fully numerical dataset with a modest feature dimension after standardization and inclusion of an intercept (Table 1). In this setting, all three solvers achieve similar validation and test losses, but they differ substantially in *optimization efficiency*, especially in the number of outer iterations required to reach a loss plateau under the common stopping rules and iteration budget.

Optimization efficiency. Table 2 reports efficiency statistics averaged across five random seeds. Both GD and Adam consistently reach the outer-iteration cap (300 iterations), indicating that neither the gradient-norm threshold nor the stagnation rule is triggered earlier under their validation-selected learning rates. By contrast, *Inexact GGN–CG* often terminates substantially earlier (64.60 ± 53.01 outer iterations on average). This reduction in outer iterations comes with additional linear-solve work (117.60 ± 86.19 total CG iterations), but the net effect is still favorable in wall-clock time: 0.0340 ± 0.0244 seconds for *Inexact GGN–CG* versus 0.0621 ± 0.0014 seconds (Adam) and 0.0733 ± 0.0012 seconds (GD).

In terms of validation loss, differences are relatively small: Adam achieves the lowest average validation mean BCE, while *Inexact GGN–CG* is slightly higher on average. This pattern is consistent with the interpretation that curvature-informed steps mainly improve *time-to-plateau* on this dataset, but do not necessarily deliver a systematic advantage in the best attainable validation loss under the fixed hyperparameter grids and iteration budgets.

Table 2: Optimization-efficiency summary averaged across random seeds for Pima Indians Diabetes. Validation loss is mean BCE (without the ℓ_2 penalty term).

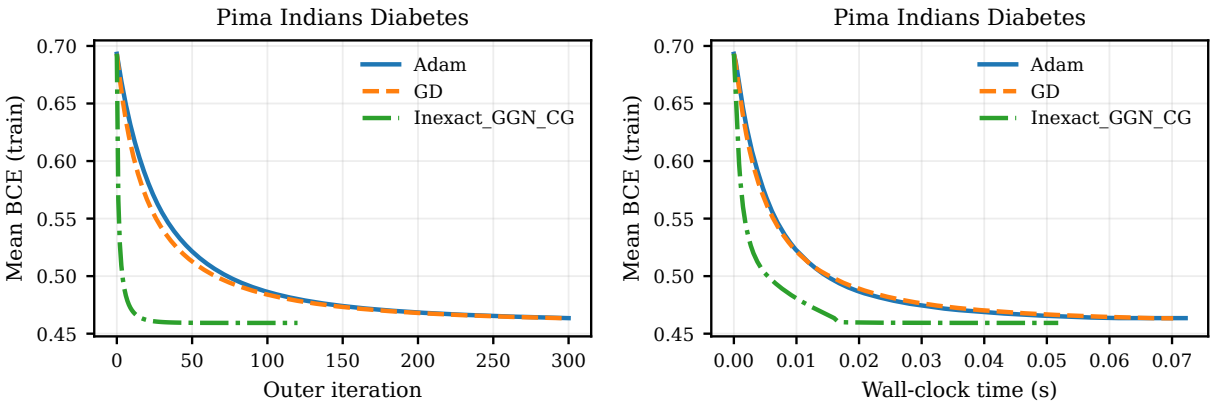
Method	Outer iters	Total CG iters	Time (s)	Val mean BCE
Adam	300.0 ± 0.00	0.0 ± 0.00	0.0621 ± 0.0014	0.4959 ± 0.0433
GD	300.0 ± 0.00	0.0 ± 0.00	0.0733 ± 0.0012	0.4969 ± 0.0419
Inexact_GGN_CG	64.6 ± 53.01	117.6 ± 86.19	0.0340 ± 0.0244	0.5002 ± 0.0441

Convergence trajectories. Figure 3 shows the mean training BCE trajectories (averaged across seeds) versus outer iteration and versus wall-clock time. *Inexact GGN–CG* exhibits a steep initial decrease and typically reaches a plateau in relatively few outer iterations, consistent with curvature-informed directions that better reflect local geometry early in optimization. GD and Adam decrease the loss more gradually and continue making incremental progress throughout the full 300-iteration budget. This qualitative behavior aligns with the cross-dataset view in Figure 2: on Pima, the additional inner effort is frequently offset by fewer outer iterations, yielding competitive total work and similar test BCE.

Generalization performance. Table 3 summarizes test-set predictive performance. All methods achieve broadly comparable accuracy and F1-score, with differences that are small relative to seed-to-seed variability. *Inexact GGN–CG* attains the lowest average test mean BCE, while GD is marginally higher on accuracy and F1-score. Overall, the primary advantage of *Inexact GGN–CG* on Pima is improved optimization efficiency (faster time-to-plateau) while maintaining competitive generalization.

Table 3: Test-set performance averaged across random seeds for Pima Indians Diabetes. Test mean BCE is computed from predicted probabilities (without the ℓ_2 penalty term). Threshold-based metrics use 0.5.

Method	Test mean BCE	Accuracy	Precision	Recall	F1-score
Adam	0.5045 ± 0.0222	0.7506 ± 0.0270	0.6578 ± 0.0428	0.5865 ± 0.0621	0.6183 ± 0.0417
GD	0.5012 ± 0.0229	0.7532 ± 0.0294	0.6671 ± 0.0573	0.5827 ± 0.0584	0.6197 ± 0.0395
Inexact_GGN_CG	0.4992 ± 0.0326	0.7455 ± 0.0240	0.6490 ± 0.0371	0.5790 ± 0.0632	0.6102 ± 0.0401



(a) Mean training BCE versus outer iteration. **(b)** Mean training BCE versus wall-clock time.

Figure 3: Pima Indians Diabetes: training-loss trajectories averaged across five random seeds.

Take-away (Pima). On a medium-scale numerical dataset, *Inexact GGN–CG* delivers substantially faster convergence in terms of outer iterations (and often time), but its improvements in validation/test BCE are modest. This indicates that the main value of curvature-informed inexact solves here lies in *reducing iteration count and reaching a plateau quickly*, rather than consistently producing a lower-loss solution under the same hyperparameter selection protocol.

5.3 Breast Cancer Wisconsin

Breast Cancer Wisconsin is a small, fully numerical dataset with strong class separability in several splits, which often yields rapid early reductions in BCE for all solvers. In such regimes, curvature information can still provide sharp descent directions, but the practical benefits may be muted because the baselines already reach low loss quickly and because the remaining improvements occur in a diminishing-returns regime where line-search and numerical conditioning effects dominate.

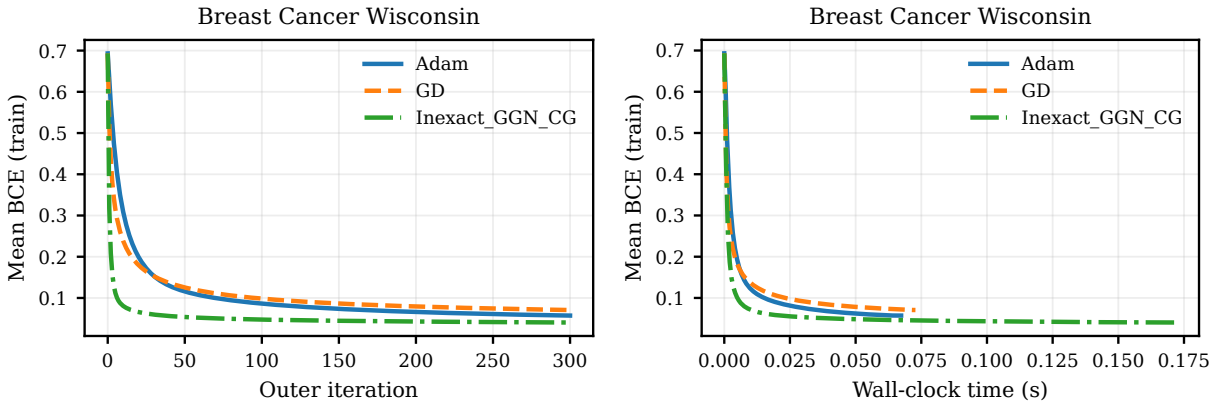
Optimization efficiency. Table 4 summarizes efficiency metrics averaged across seeds. Both GD and Adam reach the maximum outer-iteration budget (300 iterations) in every seed, indicating that their runs did not satisfy the stationarity tolerance or stagnation condition earlier under the selected learning rates. *Inexact GGN–CG* terminates earlier on average (189.8 ± 151.25 iterations), but with very high variability across seeds. This variability is accompanied by substantial and also highly variable inner linear-solve effort (342.8 ± 242.24 CG iterations) and a corresponding runtime increase (0.1074 ± 0.0805 seconds), compared with 0.0668 ± 0.0014 seconds (Adam) and 0.0787 ± 0.0039 seconds (GD).

The validation mean BCE values are close across solvers, with a slight advantage for *Inexact GGN–CG* on average (0.0885 ± 0.0456). However, the large standard deviation suggests that the validation advantage is not uniformly realized across splits. This behavior is consistent with two interacting factors: (i) near-separable splits can make the curvature term $X^\top W X$ sharply dependent on the current probability saturation (through W), and (ii) the validation-selected $(\lambda, \eta, T_{\max})$ can substantially change the conditioning of the damped system, thereby affecting the number of CG iterations and the aggressiveness of the resulting steps.

Table 4: Optimization-efficiency summary averaged across random seeds for Breast Cancer Wisconsin. Validation loss is mean BCE (without the ℓ_2 penalty term).

Method	Outer iters	Total CG iters	Time (s)	Val mean BCE
Adam	300.0 ± 0.00	0.0 ± 0.00	0.0668 ± 0.0014	0.0925 ± 0.0272
GD	300.0 ± 0.00	0.0 ± 0.00	0.0787 ± 0.0039	0.0953 ± 0.0203
Inexact_GGN_CG	189.8 ± 151.25	342.8 ± 242.24	0.1074 ± 0.0805	0.0885 ± 0.0456

Convergence trajectories. The loss trajectories in Figure 4 show that all methods rapidly attain low training BCE values. After this initial phase, further loss reduction becomes incremental, and differences between solvers are less consistently reflected in wall-clock time. In particular, while *Inexact GGN–CG* can exhibit sharper early decreases in some seeds, the overall time curve is often dominated by the accumulated cost of inner CG iterations and repeated line-search evaluations. This is a typical pattern on small, well-behaved problems: first-order solvers can be highly competitive because their iterations are cheap and the objective landscape does not strongly penalize the lack of curvature scaling.



(a) Mean training BCE versus outer iteration. **(b)** Mean training BCE versus wall-clock time.

Figure 4: Breast Cancer Wisconsin: training-loss trajectories averaged across five random seeds.

Generalization performance. Table 5 confirms that all solvers achieve strong test performance, with accuracy near 0.97 and test mean BCE around 0.09–0.10. Adam attains the best average test mean BCE and F1-score, while *Inexact GGN–CG* is slightly worse on recall and F1-score on average. In near-separable regimes, small differences in β^*, stopping time, and probability calibration can lead to noticeable changes in threshold-based metrics, even when BCE values are close. This is particularly relevant because the threshold metrics depend on calibration around 0.5, whereas BCE is sensitive to the full probability spectrum.

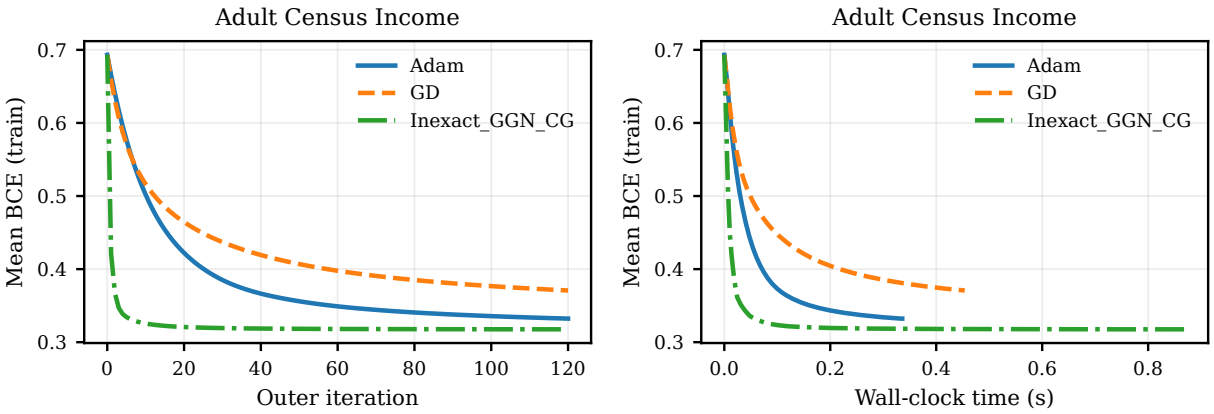
Table 5: Test-set classification performance averaged across random seeds for Breast Cancer Wisconsin. Reported loss is mean BCE with threshold 0.5.

Method	Test mean BCE	Accuracy	Precision	Recall	F1-score
Adam	0.0935 ± 0.0219	0.9737 ± 0.0107	0.9904 ± 0.0132	0.9384 ± 0.0360	0.9632 ± 0.0160
GD	0.0974 ± 0.0130	0.9702 ± 0.0100	0.9725 ± 0.0364	0.9480 ± 0.0258	0.9594 ± 0.0129
Inexact_GGN_CG	0.0997 ± 0.0298	0.9667 ± 0.0218	0.9855 ± 0.0133	0.9241 ± 0.0682	0.9524 ± 0.0339

Take-away (Breast Cancer). On a small and relatively easy dataset, first-order baselines remain highly competitive. Although *Inexact GGN–CG* can match or slightly improve validation BCE in some seeds, the additional inner CG work typically increases runtime and does not translate into consistent improvements in test BCE or threshold-based metrics. This suggests that, for small well-conditioned problems, the practical value of curvature information may be limited unless additional techniques (e.g., preconditioning or adaptive damping/forcing schedules) reduce CG effort while preserving the quality of curvature-informed steps.

5.4 Adult Census Income

Adult Census Income is the largest and most heterogeneous benchmark in our study, combining numerical and categorical predictors that produce a higher-dimensional design after one-hot encoding. This setting is practically important because the resulting feature matrix is typically



(a) Mean training BCE versus outer iteration. **(b)** Mean training BCE versus wall-clock time.

Figure 5: Adult Census Income: training-loss trajectories averaged across five random seeds.

more ill-conditioned and exhibits heterogeneous scaling and sparsity patterns, which can degrade the efficiency of poorly scaled first-order updates. Consequently, Adult Income provides a useful stress test for whether curvature-informed steps translate into measurably improved BCE minimization quality and downstream classification performance.

Optimization efficiency. Table 6 shows a clear efficiency–quality trade-off. GD and Adam both run to the outer-iteration budget of 120 in every seed, whereas *Inexact GGN–CG* terminates earlier on average (91.2 ± 49.37 outer iterations), albeit with substantial variability across seeds. The earlier termination does not imply lower cost in this case: *Inexact GGN–CG* incurs 253 ± 92.72 accumulated CG iterations, which increases wall-clock runtime to 0.6399 ± 0.2671 seconds versus 0.3555 ± 0.0063 seconds (Adam) and 0.4787 ± 0.0040 seconds (GD). Importantly, the added work yields a large validation-loss advantage: the validation mean BCE of *Inexact GGN–CG* is 0.3185 ± 0.0083, compared with 0.3321 ± 0.0063 for Adam and 0.3693 ± 0.0044 for GD. Since the outer budget for Adult Income is relatively tight (120 iterations), these differences in achieved loss within the same iteration cap are especially informative.

Table 6: Optimization-efficiency summary averaged across random seeds for Adult Census Income. Validation loss is mean BCE (without the ℓ_2 penalty term).

Method	Outer iters	Total CG iters	Time (s)	Val mean BCE
Adam	120.0 ± 0.00	0 ± 0.00	0.3555 ± 0.0063	0.3321 ± 0.0063
GD	120.0 ± 0.00	0 ± 0.00	0.4787 ± 0.0040	0.3693 ± 0.0044
Inexact_GGN_CG	91.2 ± 49.37	253 ± 92.72	0.6399 ± 0.2671	0.3185 ± 0.0083

Convergence trajectories (loss versus iteration and time). Figure 5 indicates that *Inexact GGN–CG* reaches lower training mean BCE within the iteration budget, consistent with curvature-informed directions being better scaled to the local geometry induced by mixed feature types and the sparse one-hot expansion. The time-based curves highlight the practical cost of this improvement: the additional inner CG work and line-search evaluations increase runtime. Thus, Adult Income illustrates the intended operating regime of inexact curvature methods: they can deliver a better loss value (and potentially better calibration and decision performance) at the expense of additional linear-algebra effort.

Hardware-agnostic view via a work proxy. Because wall-clock time is hardware dependent, we additionally examine loss reduction against an implementation-agnostic work proxy: the approximate number of $(Xv + X^\top u)$ products. This quantity captures the dominant linear-algebra cost for all solvers and is particularly informative for *Inexact GGN–CG*, where each

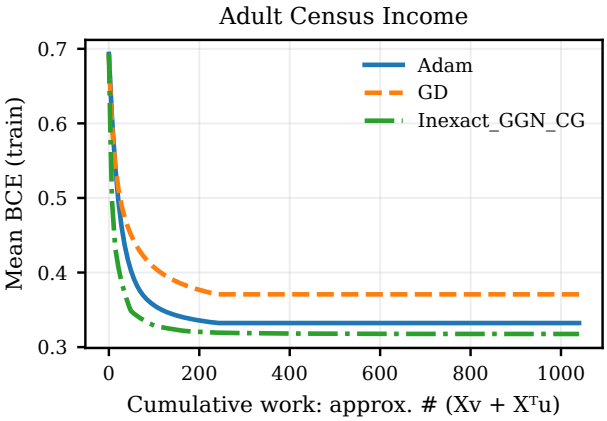


Figure 6: Adult Census Income: training mean BCE versus cumulative work (approx. number of $(Xv + X^\top u)$ products), averaged across seeds. This view factors out hardware-dependent timing and highlights how CG effort translates into loss reduction.

inner CG iteration evaluates one Xv and one $X^\top u$. Figure 6 shows that, on Adult Income, the additional CG work invested by *Inexact GGN–CG* translates into consistently lower training BCE, helping explain its superior validation and test losses. This interpretation is consistent with the cross-dataset trade-off summarized in Figure 2.

Test performance and downstream metrics. The improvements in BCE minimization quality carry over to held-out performance. Table 7 shows that *Inexact GGN–CG* achieves the lowest test mean BCE (0.3176 ± 0.0044), the highest accuracy (0.8524 ± 0.0018), and the highest F1-score (0.6623 ± 0.0066). Gains are most pronounced in recall: 0.6013 ± 0.0140 for *Inexact GGN–CG* versus 0.5612 ± 0.0123 for Adam and 0.4663 ± 0.0156 for GD, while precision remains comparable across methods. This pattern is consistent with a model that better separates positive examples without substantially increasing false positives under the fixed threshold 0.5, and it aligns with the observed reduction in BCE, which rewards well-calibrated probabilities across the full range.

Table 7: Test-set classification performance averaged across random seeds for Adult Census Income. Reported loss is mean BCE with threshold 0.5.

Method	Test mean BCE	Accuracy	Precision	Recall	F1-score
Adam	0.3323 ± 0.0018	0.8450 ± 0.0017	0.7328 ± 0.0088	0.5612 ± 0.0123	0.6355 ± 0.0060
GD	0.3694 ± 0.0015	0.8297 ± 0.0028	0.7290 ± 0.0072	0.4663 ± 0.0156	0.5686 ± 0.0118
Inexact_GGN_CG	0.3176 ± 0.0044	0.8524 ± 0.0018	0.7376 ± 0.0086	0.6013 ± 0.0140	0.6623 ± 0.0066

Take-away (Adult Income). On the largest and most heterogeneous dataset, *Inexact GGN–CG* consistently attains lower BCE and improved test metrics (particularly recall and F1) than first-order baselines under the same regularization-selection protocol. These gains come at the cost of increased inner linear-algebra work and higher runtime, but the work-proxy view indicates that the additional effort is systematically converted into objective reduction. This dataset therefore provides the strongest evidence in our study that inexact curvature solves can improve BCE minimization quality in deterministic logistic regression when the design matrix is shaped by mixed feature types and one-hot encoding.

5.5 Seed-to-seed variability and robustness

To assess robustness to data splits and to the validation-selected hyperparameters, we analyze seed-to-seed variability in both optimization-efficiency indicators and held-out loss. This analysis is particularly relevant for *Inexact GGN–CG*, which introduces additional hyperparameters

(damping λ, forcing tolerance η, and CG cap $T_{\max}$) that can materially affect conditioning, inner iteration counts, and line-search behavior. By contrast, GD and Adam typically exhibit lower variance in runtime and iteration counts under fixed budgets, especially when they frequently reach the iteration cap.

Variability in efficiency and work. Figure 7 summarizes distributions over five random seeds for Adult Income. As expected, *Inexact GGN–CG* exhibits substantially higher variability in linear-solve effort (total CG iterations) and wall-clock time, reflecting both (i) split-dependent conditioning effects and (ii) the fact that validation may select different $(\lambda, \eta, T_{\max})$ configurations across seeds. In contrast, GD and Adam show nearly degenerate distributions in outer iterations due to consistently hitting the maximum budget, and their runtime distributions are correspondingly tighter. Importantly, the higher variability in *Inexact GGN–CG* pertains primarily to *effort*, not to *stability of achieved loss*: the distribution of test mean BCE is consistently shifted downward relative to both baselines, indicating that additional work is typically translated into improved objective minimization quality.

Robustness of generalization-quality gains. The same figure indicates that, on Adult Income, the seed-wise test mean BCE of *Inexact GGN–CG* is uniformly lower than GD and, in most seeds, lower than Adam. This supports two practical conclusions. First, the method's validation-loss advantage is not driven by a single favorable split; it is observed across multiple stratified partitions. Second, while the computational cost of *Inexact GGN–CG* can vary meaningfully across seeds, the direction of the quality improvement in BCE is comparatively robust. These trends are consistent with the idea that curvature-informed directions can provide better scaling in high-dimensional heterogeneous representations, whereas the exact amount of inner work required depends on conditioning and the selected inexactness tolerance.

Implications for reporting and future improvements. The observed variability suggests that reporting only wall-clock time can be misleading when comparing curvature-based and first-order methods across hardware and implementations. Complementary reporting through a work proxy (Section 5.4) helps disentangle algorithmic effort from platform-specific timing. From a methodological standpoint, variability in total CG iterations motivates two natural extensions: (i) preconditioning to reduce CG iterations for a fixed damping level, and (ii) adaptive schedules for damping and/or forcing tolerance to reduce sensitivity to split-dependent conditioning. We do not pursue these extensions here, but the variability analysis clarifies where such improvements would most directly affect empirical performance.

5.6 Damping and inexactness: empirical implications

The proposed method introduces two coupled mechanisms that govern both stability and computational cost: *damping* of the generalized Gauss–Newton system and *inexactness* in the inner CG solve. In our protocol, the damping level λ and forcing tolerance η (together with the CG cap $T_{\max}$) are selected by validation and then held fixed within each run. Although we do not perform a dedicated hyperparameter sweep beyond the validation grids, the seed-to-seed variability in outer iterations, accumulated CG iterations, and runtime provides clear empirical evidence that these parameters materially affect performance.

Role of damping. Damping improves conditioning of the linear system

$$\left(\tfrac{1}{m} X^\top W(\theta_k) X + (\beta + \lambda) I\right) s_k = -g_k,$$

and thereby directly influences the number of CG iterations required to reach the relative residual criterion. When λ is too small, the curvature operator can be poorly conditioned (especially in

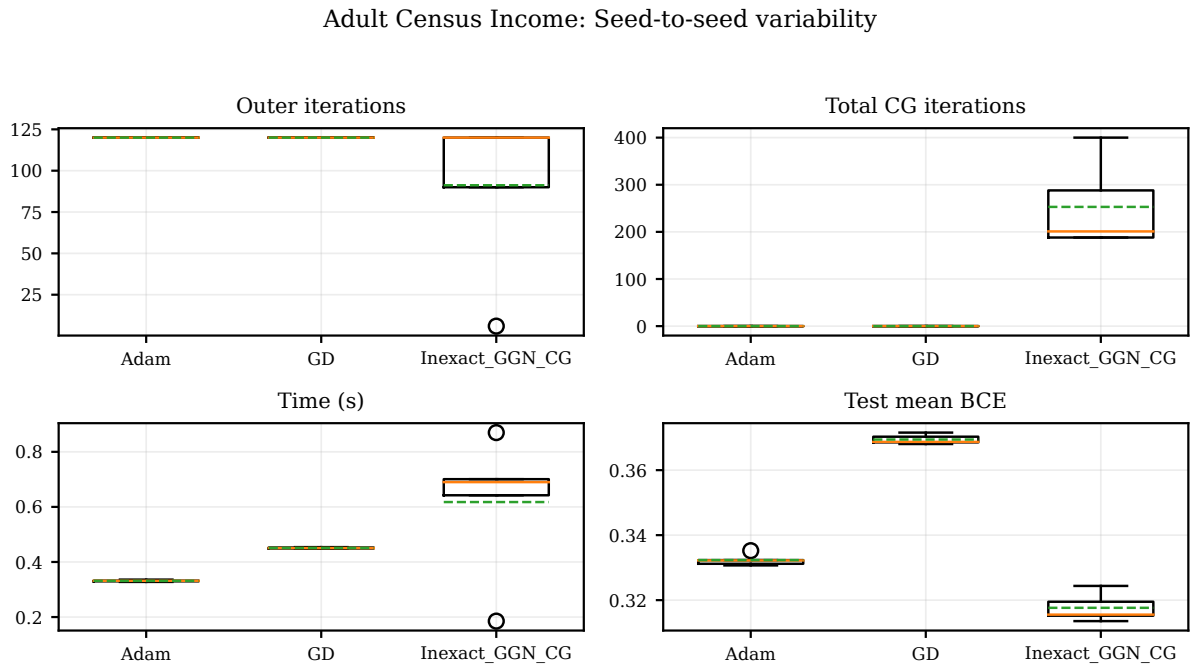


Figure 7: Adult Census Income: seed-to-seed variability across methods. Boxplots summarize outer iterations, total CG iterations, wall-clock time, and test mean BCE over five random seeds.

high-dimensional or near-collinear designs), which typically increases CG effort and may also trigger more aggressive backtracking if the approximate direction is less reliable. Conversely, overly large λ makes the system closer to $(\beta + \lambda)I$, shrinking the update and reducing the benefit of curvature information; in the extreme, the step approaches a conservative scaled-gradient update. The empirical patterns in our results are consistent with this classical trade-off: runs with larger CG totals and longer time indicate more expensive curvature solves, while runs with fewer outer iterations indicate that the curvature model is producing more effective directions.

Role of inexactness tolerance. The forcing parameter η controls how accurately CG solves the damped system through the relative residual test $\|r\|_2 \le \eta\|g\|_2$. Smaller η typically increases inner work (more CG iterations) but can reduce outer iterations by producing higher-quality curvature directions. Larger η reduces inner effort but can slow outer progress or place more burden on the line search, particularly when the approximate direction is less aligned with the true Newton/GGN direction. The existence of a clear effort–accuracy trade-off is reinforced by the work-based plots (e.g., Figure 6): on Adult Income, the additional CG work invested by *Inexact GGN–CG* translates into consistently lower BCE values, whereas on smaller and easier problems (e.g., Breast Cancer) the marginal returns of additional inner work are less pronounced.

Interaction with line search and descent safeguards. Because the direction is produced by an inexact linear solve, occasional non-descent directions can occur in finite precision or when the curvature system is solved too loosely. Our implementation enforces a descent safeguard ($s_k \leftarrow -g_k$ if $g_k^\top s_k \ge 0$) and globalizes updates via Armijo backtracking on the regularized training objective. These mechanisms ensure robustness, but they also imply that overly aggressive inexactness (large η) may lead to more conservative accepted step sizes and hence slower reduction in BCE per unit of work.

Practical take-away. Overall, the experiments suggest that *Inexact GGN–CG* is most attractive when either (i) curvature information substantially reduces outer iterations so that the added CG work is offset (as observed on Pima), or (ii) additional curvature-solve work

yields substantively improved BCE and downstream metrics within a constrained outer-iteration budget (as observed on Adult Income). On small and relatively benign problems (Breast Cancer), first-order solvers remain strong competitors, and the extra CG work does not reliably translate into better runtime or test performance.

6 Conclusion and Future Work

This paper investigated *Inexact GGN–CG*, a deterministic curvature-based method for minimizing the *mean* binary cross-entropy (BCE) objective in full-batch logistic regression. The method computes search directions by approximately solving a damped generalized Gauss–Newton system using matrix-free conjugate gradient (CG) and enforces global descent via Armijo backtracking line search on the regularized training objective. Regularization is selected once per dataset and seed and shared across solvers; validation and test losses are reported as mean BCE without the penalty term.

Experiments on three benchmark datasets reveal a clear effort–accuracy trade-off. On Pima Indians Diabetes, curvature-informed steps reduce outer iterations and reach low-loss solutions faster while maintaining competitive test performance. On Breast Cancer Wisconsin, first-order baselines remain highly competitive and the extra CG work does not consistently yield runtime or test-metric gains. On Adult Census Income, *Inexact GGN–CG* more reliably attains lower validation and test mean BCE and improves downstream recall/F1, at the cost of increased linear-algebra effort and wall-clock time.

Future work. Promising directions include adding preconditioning to reduce CG iterations, using adaptive schedules for damping and inexactness to improve robustness, and aligning solver budgets using work-based stopping criteria (e.g., a fixed budget of $(Xv + X^\top u)$ products) rather than a fixed outer-iteration cap. Extending the study to nonlinear models would further clarify when inexact GGN strategies provide the best return on computational effort.

CRediT Authorship Contribution Statement

Mohammad Jamhuri: Conceptualization, Methodology, Software, Formal Analysis, Validation, Investigation, Data Curation, Writing–Original Draft Preparation, Visualization, Project Administration. **Silvi Puspita Sari:** Methodology, Validation, Investigation, Writing–Review & Editing, Visualization. **Siti Amiroch:** Conceptualization, Supervision, Writing–Review & Editing. **Juhari:** Formal Analysis, Validation, Writing–Review & Editing. **Vivi Aida Fitria:** Resources, Data Curation, Writing–Review & Editing.

Declaration of Generative AI and AI-assisted technologies

Generative AI was used during the preparation of this manuscript. Specifically, OpenAI ChatGPT was used to assist with language editing, improving clarity and structure of the exposition, and drafting non-substantive text based on author-provided technical content. The tool was not used to generate or alter experimental data, and all algorithms, implementations, results, and interpretations were produced and verified by the authors. Figures were generated using the authors' code (Python/Matplotlib).

Declaration of Competing Interest

The authors declare no competing interests.

Funding and Acknowledgments

This research received no external funding. The authors thank their respective institutions for providing computational facilities and support. The authors also acknowledge the maintainers of the public Kaggle repositories used in this study for making the datasets available.

Data or Code Availability

The datasets analyzed during the current study are publicly available in Kaggle:[1][2][3] The code used to preprocess the data, run the experiments, and generate the figures is available from the corresponding author upon reasonable request.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2009.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[5] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[6] B. A. Pearlmutter, "Fast exact multiplication by the hessian," *Neural computation*, vol. 6, no. 1, pp. 147–160, 1994. DOI: 10.1162/neco.1994.6.1.147

[7] N. N. Schraudolph, "Fast curvature matrix-vector products for second-order gradient descent," *Neural computation*, vol. 14, no. 7, pp. 1723–1738, 2002. Available online.

[8] J. Martens et al., "Deep learning via hessian-free optimization.," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, vol. 27, 2010, pp. 735–742. Available online.

[9] A. Botev, "The gauss-newton matrix for deep learning models and its applications," Ph.D. dissertation, UCL (University College London), 2020. Available online.

[10] D. Buffelli et al., "Exact, tractable gauss-newton optimization in deep reversible architectures reveal poor generalization," *Advances in Neural Information Processing Systems*, vol. 37, pp. 133 541–133 570, 2024. DOI: 10.48550/arXiv.2411.07979

[11] J. Zhao, S. P. Singh, and A. Lucchi, "Theoretical characterisation of the Gauss–Newton conditioning in neural networks," *arXiv preprint arXiv:2502.18153*, 2024. DOI: 10.48550/arXiv.2411.02139 arXiv: 2411.02139 [cs.LG].

[12] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. Mahoney, "Adahessian: An adaptive second order optimizer for machine learning," in *proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 10 665–10 673. DOI: 10.1609/aaai.v35i12.17275

[13] H. Liu, Z. Li, D. Hall, P. Liang, and T. Ma, "Sophia: A scalable stochastic second-order optimizer for language model pre-training," *arXiv preprint arXiv:2305.14342*, 2023. DOI: 10.48550/arXiv.2305.14342Focustolearnmore

[1] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
[2] https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
[3] https://www.kaggle.com/datasets/uciml/adult-census-income

[14] D. Shin, D. Lee, J. Chung, and N. Lee, "Sassha: Sharpness-aware adaptive second-order optimization with stable hessian approximation," *arXiv preprint arXiv:2502.18153*, 2025. DOI: 10.48550/arXiv.2502.18153

[15] M. Jamhuri, M. I. Irawan, I. Mukhlash, M. Iqbal, and N. N. T. Puspaningsih, "Neural networks optimization via gauss–newton based qr factorization on sars-cov-2 variant classification," *Systems and Soft Computing*, vol. 7, p. 200 195, 2025. DOI: 10.1016/j.sasc.2025.200195

[16] M. Jamhuri, "Optimasi model deep learning menggunakan metode gauss-newton terdistribusi untuk prediksi mutasi sekuen protein spike virus sars-cov-2," Ph.D. dissertation, Institut Teknologi Sepuluh Nopember, 2025.

[17] M. Jamhuri, I. Mukhlash, and M. I. Irawan, "Performance improvement of logistic regression for binary classification by gauss-newton method," in *Proceedings of the 2022 5th International Conference on Mathematics and Statistics*, 2022, pp. 12–16. DOI: 10.1145/3545839.3545842

[18] X. Li, S. Wang, and Z. Zhang, "Do subsampled newton methods work for high-dimensional data?" In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 4723–4730. DOI: 10.1609/aaai.v34i04.5905