A COMPUTATIONALLY EFFICIENT SPARSIFIED ONLINE NEWTON METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Second-order methods have enormous potential in improving the convergence of deep neural network (DNN) training, but are prohibitive due to their large memory and compute requirements. Furthermore, computing the matrix inverse or the Newton direction, which is needed in second-order methods, requires high precision computation for stable training as the preconditioner could have a large condition number. This paper provides a first attempt at developing computationally efficient sparse preconditioners for DNN training which can also tolerate low precision computation. Our new Sparsified Online Newton (SONew) algorithm emerges from the novel use of the LogDet matrix divergence measure; we combine it with sparsity constraints to minimize regret in the online convex optimization framework. Our mathematical analysis allows us to reduce the condition number of our sparse preconditioning matrix, thus improving the stability of training with low precision. We conduct experiments on a feed-forward neural-network autoencoder benchmark, where we compare training loss of optimizers when run for a fixed number of epochs. In the float32 experiments, our methods outperform the best-performing first-order optimizers and perform comparably to Shampoo, a state-of-the-art second-order optimizer. However, our method is even more effective in low precision, where SONew finishes training considerably faster while performing comparably with Shampoo on training loss.

1 INTRODUCTION

Stochastic first order methods which use the negative gradient direction to update parameters have become the standard for training deep neural networks (DNNs). Gradient-based preconditioning involves finding an update direction, by multiplying the gradient with a preconditioner matrix carefully chosen from gradients observed in previous iterations, to improve convergence. (Full-matrix) Adagrad (Duchi et al., 2011b), online Newton method (Hazan et al., 2007) and natural gradient descent (Amari, 1998) use a full-matrix preconditioner, but computing and storing the full matrix is infeasible when there are millions of parameters. Thus, diagonal versions such as diagonal Adagrad, Adam (Kingma & Ba, 2014), and RMSprop (Hinton et al., 2012) are now widely used to train DNNs due to their scalability.

Several higher-order methods have previously been applied to deep learning (Gupta et al., 2018; Anil et al., 2020; Goldfarb et al., 2020; Martens & Grosse, 2015). All these methods use Kronecker products that reduce computational and storage costs to make them feasible for training neural networks. However, these methods rely on matrix inverses or pth-roots that require high precision arithmetic as the matrices they deal with can have large condition numbers (Anil et al., 2020; 2022). Meanwhile, deep learning hardware accelerators have evolved towards using lower precision (bfloat16, float16, int8) (Henry et al., 2019; Jouppi et al., 2017) to reduce overall computational and memory costs and improve training performance. This calls for further research in developing efficient optimization techniques that work with low precision. Indeed there is recent work along these directions, from careful quantization of Adam (Dettmers et al., 2021) to 8-bits to optimizer agnostic local loss optimization (Amid et al., 2022) that leverage first-order methods to match higher-order methods.

In this paper, we present a first attempt towards computationally efficient *sparse* preconditioners for DNN training. Regret analysis when using a preconditioner reveals that the error is bounded by two summations (see (3) below); the first summation depends on the change in the preconditioning matrix, while the second depends on the generalized gradient norm. We take the approach

of minimizing the second term while regularizing two successive preconditioners to be close in the LogDet matrix divergence measure (Kulis et al., 2009). This technique gives us an Online Newton method (Hazan et al., 2007). To make it computationally efficient, we further sparsify the preconditioner by finding a sparse approximation that is close in LogDet divergence. Thus we are consistent in using the same measure (LogDet divergence) in both the regularization and sparsification steps. This gives us our Sparsified Online Newton (SONew) method, which only requires $\mathcal{O}(n)$ time and memory complexity per iteration. We achieve this by imposing structured sparsity, such as tridiagonal and banded sparsity patterns, in the preconditioner. This is unlike most existing online Newton methods that require at least $\mathcal{O}(n^2)$ space and time complexity. By making each step linear time, the SONew method can be applied to train modern DNNs. Further, for some sparsity structures, our method is easily parallelized thus making negligible the overhead of computing the preconditioner. We also show that introducing sparsity allows us to reduce the condition number of the problem; as a consequence our preconditioner allows us to train DNNs even in low precision arithmetic.

We establish regret bound guarantees of our algorithm in the online convex optimization framework. This involves using various properties about LogDet divergence and connections to other Bregman matrix divergences (Bregman, 1967), such as the von Neumann matrix divergence (Kulis et al., 2009). We conduct experiments on an MLP Autoencoder, where we obtain better training loss compared to first order methods. We also conduct experiments on large-scale benchmarks in Appendix A.5, and observe comparable or improved performance than Adam (Kingma & Ba, 2014). Our MLP experiments in limited precision arithmetic (bfloat16) showed comparable performance with second-order methods, while being considerably faster.

2 BACKGROUND

The inner product between matrices is defined as $\langle A, B\rangle = \mathrm{Tr}(A^T B)$, where $\mathrm{Tr}(.)$ denotes the matrix trace. The Frobenius norm of a matrix A is $\|A\|_F = \sqrt{\mathrm{Tr}(A^T A)}$, while its spectral norm is $\|A\|_2 = \max_x \|Ax\|_2 / \|x\|_2$. We use $I_n \in \mathbb{R}^{n\times n}$ to denote an identity matrix. We use S_n, S_n^{++} to denote the set of symmetric, and positive definite matrices respectively. The generalized norm of a vector $x \in \mathbb{R}^n$ with respect to matrix $A \in S_n^{++}$ is defined as $\|x\|_A = \sqrt{x^T A x}$. We use $\det(A)$ to denote the determinant of matrix A, and $\mathrm{diag}(A)$ to denote diagonal matrix with $\mathrm{diag}(A)_{ii} = A_{ii}$. We use $\mathcal{G}$ and $\tilde{\mathcal{G}}$ to denote a graph and its sub-graph with a vertex set $[n] = \{1, \ldots, n\}$. Let $E_{\mathcal{G}}$ denote set of edges in graph $\mathcal{G}$, and $\mathrm{neig}_{\mathcal{G}}(i)$ denote neighbours of vertex i in graph $\mathcal{G}$. The sparsity graph/pattern of a matrix $A \in \mathbb{R}^{n\times n}$ is a graph $\mathcal{G}$ with edges $E_{\mathcal{G}} = \{(i, j) : A_{ij} \neq 0\}$ corresponding to the non-zero entries in A. We extensively use $S_n(\mathcal{G})^{++}$ to denote the set of positive definite matrices with sparsity structure given by graph $\mathcal{G}$. Given an index set $I = \{i_1, i_2, .., i_n\}$, we use A_{II} to denote the corresponding principal sub-matrix of A.

2.1 LOGDET MATRIX DIVERGENCE

Let $\phi : S_n^{++} \to \mathbb{R}$ be a strictly convex, differentiable function. Then the Bregman matrix divergence between $X, Y \in S_n^{++}$ is defined as (Bregman, 1967; Kulis et al., 2009):

$$\mathrm{D}_\phi(X, Y) = \phi(X) - \phi(Y) - \mathrm{Tr}(\nabla\phi(Y)^T (X - Y)).$$

Since ϕ is convex, $\mathrm{D}_\phi(X, U) \geq 0$ for all $X, Y \succ 0$. A well known example is when $\phi(X) = \|X\|_F^2$, the corresponding Bregman divergence $\mathrm{D}_\phi(X, Y) = \|X - Y\|_F^2$ is the squared Frobenius norm. In this paper, we extensively use the divergence when the convex function is $\phi(X) = -\log\det(X)$; the corresponding divergence measure $\mathrm{D}_{\ell\mathrm{d}}(X, Y)$ is called the *LogDet matrix divergence*:

$$\mathrm{D}_{\ell\mathrm{d}}(X, Y) = -\log\det\left(XY^{-1}\right) + \mathrm{Tr}(XY^{-1}) - n. \tag{1}$$

The LogDet divergence is scale invariant to invertible matrices A, i.e. $\mathrm{D}_{\ell\mathrm{d}}(A^T X A, A^T Y A) = \mathrm{D}_{\ell\mathrm{d}}(X, Y)$. The following revealing form of LogDet divergence in terms of eigendecompositions of $X = V\Lambda V^T$ and $Y = U\Theta U^T$ (Kulis et al., 2009):

$$\mathrm{D}_{\ell\mathrm{d}}(X, Y) = \sum_i \sum_j (v_i^T u_j)^2 (\lambda_i/\theta_j - \log(\lambda_i/\theta_j) - 1). \tag{2}$$

These two properties are later used in Section 3 to highlight the significance of LogDet divergence in our algorithm.

3 SONEW: SPARSIFIED ONLINE NEWTON METHOD

We now present our proposed SONew algorithm.

3.1 REGRET MINIMIZATION VIA LOGDET DIVERGENCE

We set up our problem under the online convex optimization framework (OCO) (Shalev-Shwartz et al., 2012; Hazan et al., 2016), where at each round the learner makes a prediction w_t in an online fashion and receives a convex loss $f_t(w_t)$ and gradient $g_t = \nabla f_t(w_t)$ as feedback. The goal of the learner is to reduce regret R_T by predicting w_t so that a low aggregate loss $\sum_{t=1}^T f_t(w_t)$ is achieved compared to the best possible, $w^* = \arg\min_w \sum_{t=1}^T f_t(w)$. Formally, the regret is given by

$$R_T(w_1, \dots, w_T) = \sum_{t=1}^T f_t(w_t) - \sum_{t=1}^T f_t(w^*).$$

To upper bound this regret, we proceed as in Hazan et al. (2016) by analyzing the error in the iterates for the update $w_{t+1} := w_t - \eta X_t g_t$, where $X_t \in \mathbb{R}^{n\times n}$. Then $\|w_{t+1} - w^*\|^2_{X_t^{-1}} = \|w_t - \eta X_t g_t - w^*\|^2_{X_t^{-1}} = \|w_t - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t - 2\eta(w_t - w^*)^T g_t$. The convexity of f_t implies that $f_t(w_t) - f_t(w^*) \le (w_t - w^*)^T g_t$ leading to $f_t(w_t) - f_t(w^*) \le \|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t$. Summing over all $t \in [T]$ and rearranging reveals the following upper bound on overall regret:

$$R_T \le \frac{1}{2\eta}\|w_1 - w^*\|^2_{X_1^{-1}} + \frac{1}{2\eta}\sum_{t=2}^T (w_t - w^*)^T (X_t^{-1} - X_{t-1}^{-1})(w_t - w^*) + \frac{\eta}{2}\sum_{t=1}^T g_t^T X_t g_t. \quad (3)$$

Since w^* is unknown, finding X_t which minimizes (3) is infeasible. So to minimize regret, we attempt to minimize the last term in (3) while regularizing X_t^{-1} to be "close" to X_{t-1}^{-1}. The nearness measure we choose is the LogDet matrix divergence, thus leading to the following objective

$$X_t = \arg\min_{X \in S_n^{++}} g_t^T X g_t, \quad \text{such that} \quad \mathrm{D}_{\ell\mathrm{d}}(X, X_{t-1}) \le c_t, \quad (4)$$

where $\mathrm{D}_{\ell\mathrm{d}}$ is as in (1). Why do we use the LogDet divergence? From (2), due to the term λ_i/θ_j, $\mathrm{D}_{\ell\mathrm{d}}(X, X_{t-1})$ prioritizes matching the smaller eigenvalues of X_{t-1} with those of X, i.e., matching the larger eigenvalues of X_{t-1}^{-1} and X^{-1}. As a consequence, LogDet divergence regularizes X by matching up its large eigenvalues with those of X_{t-1}. For e.g., if smallest and largest eigenvalue of X_{t-1} are θ_n and θ_1, then for an eigenvalue λ of X, when $\lambda > \theta_n,\ \theta_1$, the penalty from (2) for θ_n is higher than for θ_1, $(\lambda/\theta_n - \log(\lambda/\theta_n) - 1) > (\lambda/\theta_1 - \log(\lambda/\theta_1) - 1)$. This intuition leads us to formulate (4) as our objective. We recall that there is precedence of using the LogDet divergence in the optimization literature; indeed the celebrated BFGS algorithm (Broyden, 1967; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970) can be shown to be the unique solution obtained when the LogDet divergence between successive preconditioners, subject to a secant constraint, is minimized (as shown in the beautiful 4-page paper by Fletcher (1991)).

The optimization problem in (4) is convex in X since the LogDet divergence is convex in its first argument. The Lagrange $\mathcal{L}(X, \lambda_t) = g_t^T X g_t + \lambda_t(\mathrm{D}_{\ell\mathrm{d}}(X, X_{t-1}) - c_t) = \mathrm{Tr}(X g_t g_t^T) + \lambda_t(-\log\det\left(X X_{t-1}^{-1}\right) + \mathrm{Tr}(X X_{t-1}^{-1}) - n))$. Setting $\nabla\mathcal{L}(X, \lambda_t) = 0$, and using the fact that $\nabla \log\det(X) = X^{-1}$ we get the following update rule:

$$X_t^{-1} = X_{t-1}^{-1} + g_t g_t^T/\lambda_t. \quad (5)$$

Note that setting $c_t = 0$ (equivalently $\lambda_t = \infty$) $\forall t \in [n]$ in (4) results in no change to the preconditioner in any round. In this case, with $X_0 = I_n$, we get online gradient descent (Zinkevich, 2003). On the other hand, setting $\lambda_t = 1$ gives the update rule of the online Newton method Hazan et al. (2007). Our update rule differs from (full-matrix) Adagrad (Duchi et al., 2011b) which has $X_t^{-2} = X_{t-1}^{-2} + g_t g_t^T$.

Maintaining and updating X_t as in (5) is possible by using Sherman-Morrison formula but requires $\mathcal{O}(n^2)$ storage and time complexity. This becomes impractical when n is in the order of millions which is typically the case in DNNs.

Algorithm 1 Sparsified Online Newton (SONew) Algorithm

Inputs: $\lambda_t :=$ coefficient in the update (5),
$\mathcal{G} :=$ sparsity graph (banded/tridiagonal),
$\epsilon :=$ damping parameter,
$T :=$ total number of iterations/mini-batches,
$\eta_t :=$ step size/learning rate.
Output: w_{T+1}

1: $H_0 = \epsilon I_d$, $w_1 = 0$
2: **for** $t \in \{1, \ldots, T\}$ **do**
3: compute $g_t = \nabla f_t(w_t)$
4: $H_t := H_{t-1} + P_{\mathcal{G}}(g_t g_t^T/\lambda_t) \in S_n(\mathcal{G})$ with $P_{\mathcal{G}}$ as in (8). ▷ $\mathcal{O}(n)$ time & memory
5: Get $L, D =$ SPARSIFIED_INVERSE $(H_t, \mathcal{G})$, where $X_t = LDL^T$ solves (11).
6: Compute update $u_t = LDL^T g_t$,
7: $w_{t+1} = w_t - \eta_t u_t$
8: **end for**
9: **return** w_{T+1}

Algorithm 2 SPARSIFIED_INVERSE$(H, \mathcal{G})$ in $\mathcal{O}(n)$ flops

Inputs:$H \in S_n(\mathcal{G})$, is as (10).
$\mathcal{G} :=$ the banded graph of band size $b \ll n$
Outputs: lower triangular banded $L \in \mathbb{R}^{n\times n}$ and diagonal matrix $D \in \mathbb{R}^{n\times n}$

1: **function** SPARSIFIED_INVERSE$(H, \mathcal{G})$
2: $L := 0$, $D := 0$
3: $L_{jj} := 1, \forall j \in [n]$
4: **for** $j \in \{1, \ldots, n\}$ **do** ▷ parallelizable
5: Let H_{jI_j} and $H_{I_jI_j}$ be defined as in Section 2, where $I_j = \{j+1, \ldots, j+b\} \cap [n]$,
6: Solve for $L_{I_j j}$ in the linear system $H_{I_jI_j} L_{I_j j} = -H_{I_j j}$ ▷ $\mathcal{O}(b^3)$ time.
7: $D_{jj} := 1/(H_{jj} + H_{I_j j}^T L_{I_j j})$
8: **end for**
9: **return** L, D
10: **end function**

3.2 SPARSIFYING THE PRECONDITIONER

To reduce the memory required to maintain and update X_t using (5), we consider the following general problem: find sparse $X \succ 0$ with $\|X\|_0 \leq \alpha n$, $\alpha > 1$, such that the objective $\mathrm{D}_{\ell\mathrm{d}}(X, (X_{t-1}^{-1} + g_t g_t^T/\lambda_t)^{-1})$ is minimized. Due to the L_0-norm constraint, this is a non-convex problem, which makes it difficult to solve exactly. Since L_1-norm serves as a convex relaxation for the L_0 norm, we could use it instead, resulting in the following optimization problem also known as graphical lasso estimator (Friedman et al., 2008):

$$\min_{X \in S_n^{++}} \mathrm{D}_{\ell\mathrm{d}}\left(X, (X_{t-1}^{-1} + g_t g_t^T/\lambda_t)^{-1}\right) + \lambda \|X\|_1 .$$

The sparsity introduced by L_1-norm penalty will reduce the memory usage. However, the time taken to solve the above problem, even with the current best methods (Bollhöfer et al., 2019; Hsieh et al., 2013; Fattahi & Sojoudi, 2019; Zhang et al., 2018), can still be too large (as these methods take several minutes for a matrix of size million), making it impractical to embed in DNN training since preconditioning may need to be done after processing every mini-batch.

In this paper, we take a different direction where we use fixed sparsity pattern constraints, specified by a fixed undirected graph $\mathcal{G}$. To sparsify the solution in (5), we formulate the subproblem

$$X_t = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} \; \mathrm{D}_{\ell\mathrm{d}}\left(X, (X_{t-1}^{-1} + g_t g_t^T/\lambda_t)^{-1}\right), \tag{6}$$

where $S_n(\mathcal{G})^{++}$ denotes the set of positive definite matrices with the fixed sparsity pattern given by graph $\mathcal{G}$. Note that even for sparsification we use the LogDet measure; thus both the steps (4) and (6) use the same measure.

Algorithm 1 presents one proposed SONew method, which solves (6) using $\mathcal{O}(n)$ time and memory for banded matrices with band size b, in particular a tridiagonal matrix, corresponding to a chain graph, is a banded matrix with bandsize 1.

Maintaining $H_t \in S_n(\mathcal{G})$ in line 4. Solving the subproblem in (6) naively is impractical since X_{t-1}^{-1} is a dense matrix. However, the structure of the LogDet divergence comes to the rescue; (6) can be expanded as follows:

$$X_t = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} \; -\log\det(X) + \mathrm{Tr}(X(X_{t-1}^{-1} + g_t g_t^T/\lambda_t)). \tag{7}$$

Let us define the projection onto $S_n(\mathcal{G})$, $P_{\mathcal{G}} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ as:

$$P_{\mathcal{G}}(M)_{ij} = \begin{cases} M_{ij} & \text{if } (i,j) \in E_{\mathcal{G}}, \\ 0 & \text{otherwise} \end{cases}, \tag{8}$$

Note that the $\mathrm{Tr}(.)$ term in (7) involves only the non-zero elements of $X \in S_n(\mathcal{G})^{++}$. Hence, (7) can be written as

$$X_t = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} \; -\log\det(X) + \mathrm{Tr}(X P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t)), \tag{9}$$

Computing matrix X_{t-1}^{-1} can be avoided by analyzing optimality condition of (9). Let $g(X)$ be the objective function in (9), then the optimality condition of (9) is $P_{\mathcal{G}}(\nabla g(X)) = 0$. Expanding $g(X)$ gives

$$\begin{aligned} P_{\mathcal{G}}(X_t^{-1}) = P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t) &= P_{\mathcal{G}}(X_{t-1}^{-1}) + P_{\mathcal{G}}(g_t g_t^T/\lambda_t), \\ H_t &= H_{t-1} + P_{\mathcal{G}}(g_t g_t^T/\lambda_t). \end{aligned} \tag{10}$$

Thus we only need to maintain $H_t \in S_n(\mathcal{G})$, which is $H_t = P_{\mathcal{G}}(X_t^{-1})$. This matrix is updated as $H_t = H_{t-1} + P_{\mathcal{G}}(g_t g_t^T/\lambda_t)$, which can be done in $\mathcal{O}(|E_{\mathcal{G}}|)$ memory and time, while computing the matrix X_t^{-1} would have cost $\mathcal{O}(n^2)$. In SONew, this key observation is used to maintain H_t in line 4.

Computing X_t in line 5. Now that H_t is known at every round t, we can replace $P_{\mathcal{G}}(X_{t-1}^{-1} + g_t g_t^T/\lambda_t)$ in (9) with H_t as:

$$X_t = \underset{X \in S_n(\mathcal{G})^{++}}{\arg\min} \; -\log\det(X) + \mathrm{Tr}(X H_t). \tag{11}$$

For an arbitrary graph $\mathcal{G}$, solving this subproblem might be difficult. Theorems 1 and 2 show *embarrassingly parallelizable* explicit solutions to the subproblem 11 for tridiagonal and banded sparsity patterns. Proofs of Theorems 1 and 2 are given in Appendix A.1.

Theorem 1 (Explicit solution of (11) for tridiagonal matrix/chain graph)**.** *Let the sparsity structure $\mathcal{G}$ be a chain with edges $E_{\mathcal{G}} = \{(j, j+1) : j \in [n-1]\}$, and let $H \in S_n(\mathcal{G})$, then the solution of (11) is given by $\hat{X} = LDL^T$, where the unit lower triangular matrix L and diagonal matrix D have the following non-zero entries:*

$$L_{jj} = 1,\; L_{j+1j} = -\frac{H_{j+1j}}{H_{j+1j+1}},\;\; D_{jj}^{-1} = \left(H_{jj} - \frac{H_{j+1j}^2}{H_{j+1j+1}}\right), j \le n-1,\; D_{nn}^{-1} = H_{nn} \tag{12}$$

The time and memory complexity required to compute (12) is $\mathcal{O}(n)$. Note that the computation of (12) can be easily parallelized. This explicit solution can be generalized to banded sparsity structures with band size b.

Theorem 2 (Explicit solution of (11) for banded matrices)**.** *Let the sparsity pattern $\mathcal{G}$ be a banded matrix of band size b, i.e. for every vertex j, let $I_j = \{j+1, \ldots, j+b\}$, then edges $E_{\mathcal{G}} = (\bigcup_{j=1}^{n} \{j\} \times I_j) \cap \{(i,j) : i \le n, j \le n\}$. Then $X_t = LDL^T$ is the solution of (11) with nonzero entries of L and D defined as follows :*

$$L_{jj} = 1,\; L_{I_j j} = -H_{I_j I_j}^{-1} H_{I_j j},\;\; D_{jj}^{-1} = (H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j}),\; 1 \le j \le n. \tag{13}$$

Finding the above solution requires solving n linear systems of size b (which is small) as shown in Algorithm 2, and takes $\mathcal{O}((n-b+1)b^3)$ flops. Since $b \ll n$, the number of flops is $\mathcal{O}(n)$.

3.3 Analysis of SONew

The following theorem establishes regret guarantees of SONew in online convex optimization framework mentioned in Section 3.1.

Theorem 3. *When $\mathcal{G}$ = tridiagona/chain graph, then Algorithm 1 incurs the following regret under convex losses f_t*

$$R_T \le \mathcal{O}\left(C^{1/2} \cdot T^{3/4} \cdot \left(\frac{1+\beta}{1-\beta}\right)^{1/2} \cdot \left(\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right) \right)^{3/4} \right),$$

where $g_{1:T}^{(i)} = [(g_1)_i, \dots (g_T)_i]$ *denotes gradient history of* i^{th} *variable/parameter* ,$G_\infty^{(i)} = \|g_{1:T}^{(i)}\|_\infty$, $C = \max_t(\sum_{i=1}^n (w_t - w^*)_i^2 (G_\infty^{(i)})^2)$, $\beta_i = \frac{\left\langle g_{1:T}^{(i)}, g_{1:T}^{(i+1)} \right\rangle}{\sqrt{(\epsilon + \left\|g_{1:T}^{(i)}\right\|_2^2) \cdot (\epsilon + \left\|g_{1:T}^{(i+1)}\right\|_2^2)}}$ *denotes dot-product between gradient histories of connected parameters in the chain graph,* $\beta = \max_{i \in [n-1]} \beta_i$.

SONew is *scale-invariant* to diagonal transformation of gradients; if the gradients sent as feedback to the OCO learner are $\bar{g}_t = \Lambda g_t$, then the iterates for the transformed problem will be $\bar{x}_t = \Lambda^{-1} w_t = \Lambda^{-1} x_{t-1} - \eta(\Lambda^{-1} X_{t-1} \Lambda^{-1}) \Lambda(g_{t-1})$, this is due to scale-invariance property of LogDet divergence in Section 2.1,i.e, the transformed preconditioner is $\bar{X}_t = \Lambda^{-1} X_{t-1} \Lambda^{-1}$. Furthermore, the regret bound derived in Theorem 3 for transformed and original problem is approximately same. Our regret bound is data-dependent, since if β_i is nearer to 1, then the regret is smaller due to the $\log(1 - \beta_i^2)$ term. This effect is more amplified when $\log(1 + \|g_{1:T}^{(i)}\|_2^2/\epsilon)$ is relatively low compared to $\log(1 - \beta_i^2)$. The proof for Theorem 3 is given in Appendix A.2. We also derive a regret bound with $\mathcal{O}(\sqrt{\kappa} T^{1/2})$ dependency on T in Appendix A.3, where the condition number $\kappa(\text{diag}(H_t)) \leq \kappa$.

4 Numerical Stability of SONew

The matrix $H \in S_n(\mathcal{G})$ given as input to Algorithm 2 should have positive definite submatrices $H_{J_j J_j}$, $J_j = I_j \cup \{j\}$, $j \in [n]$, where I_j is as in Theorem 2, hence, it should have positive schur complements $(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j})$. But, if the matrices $H_{J_j J_j}$ and $H_{I_j I_j}$ are illconditioned, then the computed value for $(H_{jj} + H_{I_j j}^T L_{I_j j}) = (H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j})$, in line 7 of Algorithm 2, can be zero or negative due to catastrophic cancellation, since finding $L_{I_j j} = -H_{I_j I_j}^{-1} H_{I_j j}$ in floating point arithmetic can result in rounding errors. To understand this issue further, we conduct perturbation analysis in Theorem 4 which establishes a componentwise condition number (Higham, 2002) of the optimization problem in (11) with a tridiagonal sparsity structure $\mathcal{G}$.

Theorem 4 (Condition number of tridiagonal LogDet subproblem 11)**.** *Let* $H \in S_n^{++}$ *be such that* $H_{ii} = 1$ *for* $i \in [n]$. *Let* ΔH *be a symmetric perturbation such that* $\Delta H_{ii} = 0$ *for* $i \in [n]$, *and* $H + \Delta H \in S_n^{++}$. *Let* $P_{\mathcal{G}}(H)$ *be the input to 11, where* $\mathcal{G}$ *is a chain graph, then*

$$\kappa_\infty^{\ell d} \leq \max_{i \in [n-1]} 2/(1 - \beta_i^2) = \hat{\kappa}_\infty^{\ell d}, \tag{14}$$

where, $\beta_i = H_{ii+1}$,$\kappa_\infty^{\ell d} :=$ *componentwise condition number of (11) for perturbation* ΔH. [1]

So, the tridiagonal LogDet problem with inputs H as mentioned in Theorem 4, has high condition number when $1 - \beta_i^2 = H_{ii} - H_{ii+1}^2 / H_{i+1i+1}$ are low and as a result the preconditioner X_t in SONew (Algorithm 1) has high componentwise relative errors. In SONew (1), the $H_t = P_{\mathcal{G}}(\sum_{s=1}^t g_s g_s^T / \lambda_t)$ generated in line 4 could be such that the matrix $\sum_{s=1}^t g_s g_s^T / \lambda_t$ need not be positive definite and so the schur complements $H_{ii} - H_{ii+1}^2 / H_{i+1i+1}$ can be zero, giving an infinite condition number $\kappa_\infty^{\ell d}$ by Theorem 4. The following lemma describes such cases in detail for a more general banded sparsity structure case.

Lemma 1 (Degenerate inputs to banded LogDet subproblem)**.** *Let* $H = P_{\mathcal{G}}(GG^T)$, *where* $G \in \mathbb{R}^{n \times T}$ *and let* $g_{1:T}^{(i)}$ *be* i^{th} *row of* G, *which is gradients of parameter* i *for* T *rounds, then* $H_{ij} = \left\langle g_{1:T}^{(i)}, g_{1:T}^{(j)} \right\rangle$.

- *Case 1: For tridiagonal sparsity structure* $\mathcal{G}$*: if* $g_{1:T}^{(j)} = g_{1:T}^{(j+1)}$, *then* $H_{jj} - H_{jj+1}^2 / H_{j+1j+1} = 0$.
- *Case 2: For* $b > 1$ *in (13): If* $\text{rank}(H_{J_j J_j}) = \text{rank}(H_{I_j I_j}) = b$, *then* $(H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j}) = 0$ *and* $D_{jj} = \infty$. *If* $\text{rank}(H_{I_j I_j}) < b$ *then the inverse* $H_{I_j I_j}^{-1}$ *doesn't exist and* D_{jj} *is not well-defined.*

If $GG^T = \sum_{i=1}^T g_i g_i$ is a singular matrix, then solution to the LogDet problem might not be well-defined as shown in Lemma 1. For instance, Case 1 can occur when preconditioning the input layer

[1] The full version of Theorem 4 along with proof is given in Appendix A.4.

of an image-based DNN with flattened image inputs, where j^{th} and $(j+1)^{th}$ pixel can be highly correlated throughout the dataset. Case 2 can occur in the first b iterations in Algorithm 1 when the rank of submatrices $\text{rank}(H_{I_jI_j}) < b$ and $\epsilon = 0$.[2]. We develop Algorithm 3 which is robust to degenerate inputs H, given that $H_{ii} > 0$. It finds a subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$ for which (13) is well-defined. This is done by removing edges which causes inverse $H^{-1}_{I_jI_j}$ to be singular or $(H_{jj} - H^T_{I_jj} H^{-1}_{I_jI_j} H_{I_jj})$ to be low. Our ablation study in Table 4 demonstrates noticeable improvement in performance Algorithm 3 is used.

Theorem 5 (Numerically stable algorithm)**.** *Algorithm 3 finds a subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$, such that explicit solution for $\tilde{\mathcal{G}}$ in (13) is well-defined. Furthermore, when $\mathcal{G}$ is a tridiagonal/chain graph, the component-wise condition number upperbound in (14) is reduced upon using Algorithm 3, $\hat{\kappa}^{\tilde{\mathcal{G}}}_{\ell d} < \hat{\kappa}^{\mathcal{G}}_{\ell d}$, where $\hat{\kappa}^{\tilde{\mathcal{G}}}_{\ell d}$, $\hat{\kappa}^{\mathcal{G}}_{\ell d}$ are defined as in Theorem 4 for graphs $\tilde{\mathcal{G}}$ and $\mathcal{G}$ respectively.*

The proofs for Lemma 1 and Theorem 5 are given in Appendix A.4.

Algorithm 3 Numerically stable banded LogDet solution

1: **Input:** $\mathcal{G}-$ tridiagonal or banded graph, $H-$ symmetric matrix in $\mathbb{R}^{n\times n}$ with sparsity structure $\mathcal{G}$ and $H_{ii} > 0$, $\gamma-$ tolerance parameter for low schur complements.
2: **Output:** Finds subgraph $\tilde{\mathcal{G}}$ of $\mathcal{G}$ without any degenerate cases from Lemma 1 and finds preconditioner $\hat{X}$ corresponding to the subgraph
3: Let $E_i = \{(i,j) : (i,j) \in E_{\mathcal{G}}\}$ be edges from vertex i to its neighbours in graph $\mathcal{G}$.
4: Let $V_i^+ = \{j : i < j, (i,j) \in E_{\mathcal{G}}\}$ and $V_i^- = \{j : i > j, (i,j) \in E_{\mathcal{G}}\}$, denote positive and negative neighbourhood of vertex i.
5: Let $K = \left\{ i : H_{ii} - H^T_{I_ii} H^{-1}_{I_iI_i} H_{I_ii} \text{ is not defined or } \leq \gamma, i \in [n] \right\}$
6: Consider a new subgraph $\tilde{\mathcal{G}}$ with edges $E_{\tilde{\mathcal{G}}} = E_{\mathcal{G}} \setminus (\bigcup_{i\in K} E_i \cup (V_i^+ \times V_i^-))$
7: **return** $\hat{X} := \textsc{Sparsified_Inverse}\ (\tilde{H}_t, \tilde{\mathcal{G}})$, where $\tilde{H}_t = P_{\tilde{\mathcal{G}}}(H_t)$

5 Related Work

Online Newton method is a second order method in online convex optimization framework with properties such as scale invariance (Luo et al., 2016) and logarithmic regrets in exp-concave and strongly convex functions (Hazan et al., 2007; 2016). However, it has a time complexity of $\mathcal{O}(n^2)$, making it infeasible for large n. A diagonal version of this method SC-Adagrad (Mukkamala & Hein, 2017) was proposed to make the online Newton method scalable for deep-learning. SC-Adagrad is equivalent to setting the sparsity pattern $\mathcal{G}$ in equation 11 as a null graph, furthermore, setting $\mathcal{G}$ as a complete graph will result in online Newton method. However, introduction of LogDet divergence measure in SONew allows us to set different sparsity graphs as $\mathcal{G}$ such as banded graph with band-size b, for which our preconditioning process is more computationally efficient with a time complexity of $\mathcal{O}(b^3(n-b+1))$ compared to online-newton method $\mathcal{O}(n^2)$. As discussed in Theorem 3, SONew also utilizes correlation among gradients of parameters to improve convergence, in contrast to diagonal preconditioners such as SC-Adagrad.

Shampoo (Gupta et al., 2018; Anil et al., 2020) uses Kronecker factored preconditioners to reduce the memory and time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(d_1^2 + d_2^2)$ and $\mathcal{O}(d_1^3 + d_2^3)$ respectively, where $n = d_1 d_2$ denotes number of parameters for a linear layer of dimensions $d_1 \times d_2$. The time complexity of matrix inversion takes a heavy toll in Shampoo's compute time even with the Kronecker product assumption on the preconditioner, whereas, our method has a time complexity of $\mathcal{O}(b^3 d_1 d_2)$ quadratic in dimensions of the linear layer (note that $b = 1$ for tridiagonal structure). Furthermore, precision lost in matrix inversion can be large due to high condition number matrices occurring in training (Anil et al., 2022; 2020).

LogDet problem in equation 11 is closely related to the Maximum Determinant Matrix Completion (MDMC) (Andersen et al., 2013; Vandenberghe et al., 2015). The MDMC problem is the dual of LogDet problem (11), and has explicit solutions for chordal graphs (Andersen et al., 2013). Thus the explicit solutions in (13) are the same as the ones proved in Andersen et al. (2013). Also, we noticed that the tridiagonal explicit solution has been used previously in KFAC (Martens & Grosse, 2015) in the context of a gaussian graphical model interpretation of gradients. In this paper we also analyze conditioning and degenerate cases of these explicit solutions (which can occur frequently in

[2]This is proved in Appendix A.4

DNN training) and develop algorithms to avoid such cases. In addition, we have regret guarantees connecting these explicit solutions to regret in online convex optimization problem setup. There is prior work (Luo et al., 2016; 2019) in reducing the complexity - $\mathcal{O}(n^2)$ flops of Online Newton Step (ONS) to $\mathcal{O}(n)$ flops using sketching. These ONS variants maintain a low rank approximation of H_t (as in Algorithm 1) and updating it with a new gradient g_t at every iteration requires conducting SVD (Luo et al., 2019)/orthonormalization (Luo et al., 2016) of a tall and thin matrix in $\mathbb{R}^{n\times r}$, where r denotes the rank of approximation of H_t. Our proposed method (Algorithm 1) has a more parallelizable update $H_t \coloneqq H_{t-1} + P_{\mathcal{G}}(g_t g_t^T)$ making it more suitable for DNN training.

6 Experimental Results

We conduct experiments on the Autoencoder benchmark (Schmidhuber, 2015) using MNIST dataset (Deng, 2012). We compare SONew against commonly used first order methods including SGD (Kiefer & Wolfowitz, 1952)), SGD with Momentum (Qian, 1999), Nesterov (Nesterov, 1983), Adagrad (Duchi et al., 2011a), Adam (Kingma & Ba, 2014), and Rmsprop (Tieleman & Hinton, 2012). We also compare with Shampoo (Gupta et al., 2018), a state of the art second-order optimizer used in practice. Computing preconditioner at every step in shampoo could be infeasible, instead it is computed every t steps - referred to as Shampoo(t) in the experiments.

For SONew, we use exponentially moving average (EMA) for both first order and second order statistics. EMA is commonly used in adaptive optimizers for deep-learning training Kingma & Ba (2014); Qian (1999). Let $\beta_1, \beta_2 \in [0, 1]$ be the coefficients for EMA and $g_t \in \mathbb{R}^n$ be the gradient at t^{th} iteration/mini-batch. Let $\mu_t \in \mathbb{R}^n$ be the first order gradient statistic, and $H_t \in S_n(\mathcal{G})$ be the second order gradient statistic as in (10). Then, the following modification to update rules are made to in the SONew implementation.

$$\mu_t = \beta_1 \mu_{t-1} + (1-\beta_1) g_t, \quad H_t = \beta_2 H_{t-1} + (1-\beta_2) g_t g_t^T.$$

Furthermore, μ_t is used in $w_{t+1} = w_t - \eta_t X_t \mu_t$, replacing the gradient g_t, similar to Kingma & Ba (2014). As discussed in Section 3.2 we need to store the values of H_t only corresponding to the sparsity pattern, hence SONew uses $\mathcal{O}(n)$ space. The updates above get computed in parallel in $\mathcal{O}(1)$ time. Moreover, we use Algorithm 3 to make SONew numerically stable. We also use grafting (Agarwal et al., 2022), a technique used to transfer step size between optimization algorithms. Specifically, given an update v_1 of Optimizer-1 and v_2 of Optimizer-2, grafting allows to use the direction suggested by Optimizer-2 with step size suggested by Optimizer-1. The final update is given by $\frac{\|v_1\|}{\|v_2\|} \cdot v_2$. Grafting has been shown to take advantage of a tuned optimizer step size and improve performance. In our case, we use Adam grafting - using Adam optimizer step size $\|v_1\|$ with SONew direction $v_2/\|v_2\|$.

We use three sparsity patterns for SONew - a) diagonal sparsity, resulting in a diagonal preconditioner similar to adaptive first order methods like Adam and Adagrad; b) tridiagonal sparsity, corresponding to a chain graph; and c) banded sparsity, represented by "band-k" in tables and figures for band size of k.

All the baselines and SONew are trained for 100 epochs and use the train set of MNIST dataset containing $60k$ points. We use a standard sized (2.72M parameters) Autoencoder with layer sizes: [1000, 500, 250, 30, 250, 500, 1000] and tanh non-linearity. Batch size is fixed at 1000 and we use learning rate schedule with a linear warmup of 5 epochs followed by linear decay towards 0. Over $2k$ hyperparameters are searched over for each experiment using a Bayesian optimization package. The search space for each optimizer is mentioned in Appendix A.5.

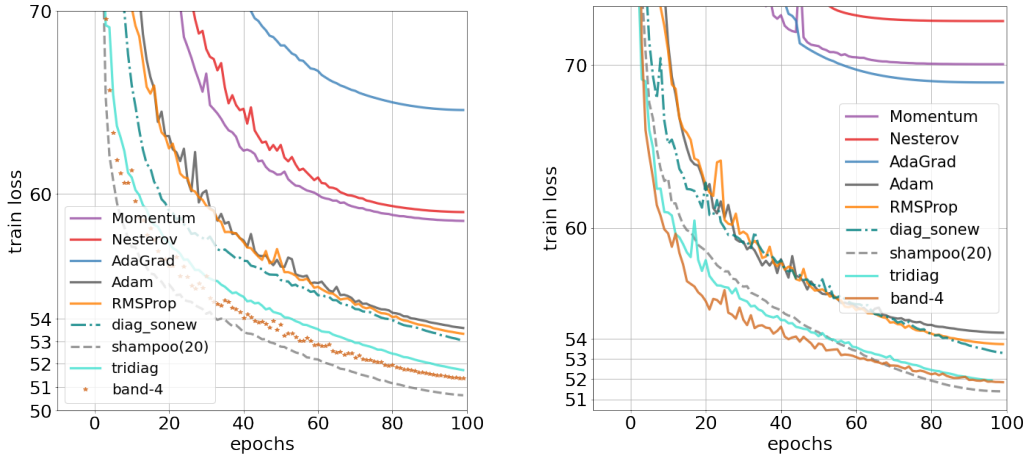
From the float32 experiments in Table 1 we observe that among first order methods, diag-SONew performs the best while taking same amount of time. Increasing the number of edges in the sparsity graph to tridiag or banded with band size 4 enhances the performance further. Tridiag-SONew performs $4\times$ faster than Shampoo at a marginal cost to the loss - even when Shampoo updates preconditioner once every 20 steps. When dealing with large scale models like Resnet50 (He et al., 2015), Shampoo requires updating preconditioner much more often (Anil et al., 2019), whereas our method is also promising in such scenarios. We leave testing on such large benchmarks as a future work. To show efficacy of SONew at lower precision, we conduct bfloat16 experiments. We notice in Table 2 that diag-SONew performs the best in first order methods, just like in float32. Moreover SONew undergoes the least degradation in performance compared to all other optimizers. We also provide an ablation study on effect of using Algorithm 3 on training loss in the appendix.

Table 1: **float32 experiments on Autoencoder benchmark.** "tridiag" repersents tridiag-SONew, and "band-4" represents banded-SONew with band size 4. We observe that diag-SONew performs the best among all first order methods while taking similar time. tridiag and band-4 perform significantly better than first order methods while requiring similar linear space and being marginally slower. Shampoo performs best but takes $\mathcal{O}(d^3)$ time for computing preconditioner of a linear layer of size $d \times d$, whereas our methods take $\mathcal{O}(d^2)$ to find the precondioner.

Optimizer	**First Order Methods**							**Second Order Methods**		
	SGD	**Nesterov**	**Adagrad**	**Momentum**	**RMSProp**	**Adam**	**diag-SONew**	**Shampoo(20)**	**tridiag**	**band-4**
Train CE loss	67.654	59.087	54.393	58.651	53.330	53.591	53.025	50.702	51.723	51.357
Time(s)	62	102	62	67	62	62	63	371	87	251

Table 2: **bfloat16 experiments on Autoencoder benchmark.** diag-SONew performs the best among all first order methods, while degrading only marginally (0.26 absolute difference) compared to float32 performance. tridiag-SONew holds similar observation as well, with only 0.35 increase in loss. Shampoo performs the best but has a considerable drop (0.70) in performance compared to float32 due to using matrix inverse, and is slower due to its cubic time complexity for computing preconditioners. Shampoo implementation uses 16-bit quantization to make it work in 16-bit setting, leading to further slowdown. Hence the running time in bfloat16 is even more than in float32.

Optimizer	**First Order Methods**							**Second Order Methods**		
	SGD	**Nesterov**	**Adagrad**	**Momentum**	**RMSProp**	**Adam**	**diag-SONew**	**Shampoo(20)**	**tridiag**	**band-4**
Train CE loss	80.454	72.975	68.854	70.053	53.743	54.328	53.29	51.401	51.937	51.84
Train time(s)	36	43	37	36	37	38	44	1245	70	502



(a) float32 (b) bfloat16

Figure 1: (a) Comparison of SONew (tridiag, band-4) with first-order optimizers and Shampoo (second-order). Left (a) uses float32 training and right (b) uses bfloat16 training. We observe that tridiag-SONew has better convergence compared to shampoo in lower precision in early stages, and performs better than all first order methods in float32 and bfloat16.

7 Conclusions and Future Work

In this paper we have introduced a computationally efficient sparse preconditioner. Our algorithm arises from a novel regret bound analysis using LogDet Divergence, and furthermore we make it numerically stable. Experimental results on the Autoencoder benchmark confirm the effectiveness of SONew in both float32 as well as bfloat16 precision. In the future, one can test SONew on larger benchmarks like DLRM (Naumov et al., 2019) and ResNet50 (He et al., 2015), and explore different sparsity graphs for which efficient solutions exist for the LogDet subproblem (11).

References

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Learning rate grafting: Transferability of optimizer tuning, 2022. URL https://openreview.net/forum?id=FpKgG31Z_i9.

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

Ehsan Amid, Rohan Anil, and Manfred Warmuth. Locoprop: Enhancing backprop via local loss optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 9626–9642. PMLR, 2022.

Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. Logarithmic barriers for sparse matrix cones. *Optimization Methods and Software*, 28(3):396–423, 2013.

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory-efficient adaptive optimization, 2019. URL https://arxiv.org/abs/1901.11150.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

Rohan Anil, Sandra Gadanho, Da Huang, Nijith Jacob, Zhuoshu Li, Dong Lin, Todd Phillips, Cristina Pop, Kevin Regan, Gil I Shamir, et al. On the factory floor: ML engineering for industrial-scale ads recommendation models. *arXiv preprint arXiv:2209.05310*, 2022.

Matthias Bollhöfer, Aryan Eftekhari, Simon Scheidegger, and Olaf Schenk. Large-scale sparse inverse covariance matrix estimation. *SIAM Journal on Scientific Computing*, 41(1):A380–A401, 2019.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

Charles G Broyden. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.

Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011a. URL http://jmlr.org/papers/v12/duchi11a.html.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011b.

Salar Fattahi and Somayeh Sojoudi. Graphical lasso and thresholding: Equivalence and closed-form solutions. *Journal of machine learning research*, 2019.

Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

Roger Fletcher. A new variational result for quasi-Newton formulae. *SIAM Journal on Optimization*, 1(1):18–21, 1991.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-Newton methods for training deep neural networks. *Advances in Neural Information Processing Systems*, 33:2386–2396, 2020.

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

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learningfor image recognition. *ComputerScience*, 2015.

Greg Henry, Ping Tak Peter Tang, and Alexander Heinecke. Leveraging the bfloat16 artificial intelligence datatype for higher-precision computations. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 69–76. IEEE, 2019.

Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

Cho-Jui Hsieh, Mátyás A Sustik, Inderjit S Dhillon, Pradeep K Ravikumar, and Russell Poldrack. BIG & QUIC: Sparse inverse covariance estimation for a million variables. *Advances in neural information processing systems*, 26, 2013.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017.

J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952. doi: 10.1214/aoms/1177729392. URL https://doi.org/10.1214/aoms/1177729392.

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

Brian Kulis, Mátyás A Sustik, and Inderjit S Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10(2), 2009.

G Lindblad. Completely positive maps and entropy inequalities. 1975.

Haipeng Luo, Alekh Agarwal, Nicolo Cesa-Bianchi, and John Langford. Efficient second order online learning by sketching. *Advances in Neural Information Processing Systems*, 29, 2016.

Luo Luo, Cheng Chen, Zhihua Zhang, Wu-Jun Li, and Tong Zhang. Robust frequent directions with application in online learning. *The Journal of Machine Learning Research*, 20(1):1697–1737, 2019.

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

Mahesh Chandra Mukkamala and Matthias Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. In *International conference on machine learning*, pp. 2545–2553. PMLR, 2017.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(1/k^2). 1983.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(98)00116-6. URL https://www.sciencedirect.com/science/article/pii/S0893608098001166.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, jan 2015. doi: 10.1016/j.neunet.2014.09.003. URL https://doi.org/10.1016%2Fj.neunet.2014.09.003.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

David F Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

T. Tieleman and G. Hinton. Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. 2012.

Lieven Vandenberghe, Martin S Andersen, et al. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization*, 1(4):241–433, 2015.

Richard Zhang, Salar Fattahi, and Somayeh Sojoudi. Large-scale sparse inverse covariance estimation via thresholding and max-det matrix completion. In *International Conference on Machine Learning*, pp. 5766–5775. PMLR, 2018.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

A Appendix

A.1 Properties of LogDet subproblem

Proof of Theorem 2

The optimality condition of (11) is

$$P_{\mathcal{G}}(X^{-1}) = P_{\mathcal{G}}(H)$$

Let $Z = L^{-T}D^{-1}L^{-1}$, then $P_{\mathcal{G}}(Z) = H$

$$ZL = L^{-T}D^{-1} \implies ZLe_j = L^{-T}D^{-1}e_j$$

Let $J_j = I_j \cup j$, then select J_j indices of vectors on both sides of the second equality.

$$\begin{bmatrix} Z_{jj} & Z_{jI_j} \\ Z_{I_jj} & Z_{J_jJ_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{I_j} \end{bmatrix} = \begin{bmatrix} 1/d_{jj} \\ 0 \end{bmatrix} \tag{15}$$

Note that L^{-T} is an upper triangular matrix with ones in the diagonal hence J_j^{th} block of $L^{-T}e_j$ will be $[1, 0, 0, \ldots]$. Also, since $P_{\mathcal{G}}(Z) = H$

$$\begin{bmatrix} Z_{jj} & Z_{jI_j} \\ Z_{I_jj} & Z_{J_jJ_j} \end{bmatrix} = \begin{bmatrix} H_{jj} & H_{jI_j} \\ H_{I_jj} & H_{J_jJ_j} \end{bmatrix}$$

Substituting this in the linear equation 15

$$\begin{bmatrix} H_{jj} & H_{jI_j} \\ H_{I_jj} & H_{J_jJ_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{I_j} \end{bmatrix} = \begin{bmatrix} 1/d_{jj} \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} H_{jj} & H_{jI_j} \\ H_{I_jj} & H_{J_jJ_j} \end{bmatrix} \begin{bmatrix} d_{jj} \\ d_{jj} \cdot L_{I_j} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$H_{jj}d_{jj} + d_{jj}H_{I_jj}^T L_{I_jj} = 1$$

$$H_{I_jj}d_{jj} + d_{jj}H_{I_jI_j}L_{I_jj} = 0$$

The lemma follows from solving the above equations. Note that here we used that lower triangular halves of matrices L and H have the same sparsity patterns, which follows from the fact that banded graph is chordal and has a perfect elimination order $[1, 2, \ldots, n]$.

Proof of Theorem 1 The proof follows trivially from Theorem 1, when b is set to 1.

A.2 REGRET BOUND ANALYSIS

Proof of Theorem 3.

To upperbound the regret we prove the following lemma about regret decompsition

Lemma 2 (Hazan et al. (2016)). *In the OCO problem setup, if a prediction $w_t \in \mathbb{R}^n$ is made at round t and is updated as $w_{t+1} := w_t - \eta X_t g_t$ using a preconditioner matrix $X_t \in S_n^{++}$*

$$R_T \leq \frac{1}{2\eta} \cdot (\|w_1 - w^*\|^2_{X_1^{-1}} - \|w_{T+1} - w^*\|_{X_T^{-1}}) \tag{16}$$

$$+ \frac{1}{2\eta} \cdot \sum_{t=1}^{T-1} (w_{t+1} - w^*)^T (X_{t+1}^{-1} - X_t^{-1})(w_{t+1} - w^*) \tag{17}$$

$$+ \sum_{t=1}^{T} \frac{\eta}{2} \cdot g_t^T X_t g_t \tag{18}$$

Proof.

$$\begin{aligned} \|w_{t+1} - w^*\|^2_{X_t^{-1}} &= \|w_t - \eta X_t g_t - w^*\|^2_{X_t^{-1}} \\ &= \|w_t - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t - 2\eta (w_t - w^*)^T g_t \\ \implies 2\eta (w_t - w^*)^T g_t &= \|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} + \eta^2 g_t^T X_t g_t \end{aligned}$$

□

Using the convexity of f_t, $f_t(w_t) - f_t(w^*) \leq (w_t - w^*)^T g_t$, where $g_t = \Delta f_t(w_t)$ and summing over $t \in [T]$

$$R_T \leq \sum_{t=1}^{T} \frac{1}{2\eta} \cdot \left(\|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} \right) + \frac{\eta}{2} \cdot g_t^T X_t g_t \tag{19}$$

The first summation can be decomposed as follows

$$\begin{aligned} \sum_{t=1}^{T} \left(\|w_t - w^*\|^2_{X_t^{-1}} - \|w_{t+1} - w^*\|^2_{X_t^{-1}} \right) &= \left(\|w_1 - w^*\|^2_{X_1^{-1}} - \|w_{T+1} - w^*\|^2_{X_T^{-1}} \right) \\ &+ \sum_{t=1}^{T-1} (w_{t+1} - w^*)^T (X_{t+1}^{-1} - X_t^{-1})(w_{t+1} - w^*) \end{aligned}$$

Substituting the above identity in the Equation (19) proves the lemma.

Let $R_T \leq T_1 + T_2 + T_3$, where

- $T_1 = \frac{1}{2\eta} \cdot (\|w_1 - w^*\|^2_{X_1^{-1}} - \|w_{T+1} - w^*\|_{X_T^{-1}})$
- $T_2 = \frac{1}{2\eta} \cdot \sum_{t=1}^{T-1} (w_{t+1} - w^*)^T (X_{t+1}^{-1} - X_t^{-1})(w_{t+1} - w^*)$
- $T_3 = \sum_{t=1}^{T} \frac{\eta}{2} \cdot g_t^T X_t g_t$

The following lemmas will be used to bound T_1, T_2, T_3.

Lemma 3. *If $\mathcal{G}$ = chain/tridiagonal graph and $\hat{X} = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, H^{-1})$, then the inverse $\hat{X}^{-1}$ takes the following expression*

$$(\hat{X}^{-1})_{ij} = \begin{cases} H_{ij} & |i-j| \leq 1 \\ \frac{H_{ii+1} H_{i+1i+2} \ldots H_{j-1j}}{H_{i+1i+1} \ldots H_{j-1j-1}} & \end{cases} \tag{20}$$

Proof.

$$\hat{X}^{-1}\hat{X}^{(j)} = e_j$$

Where $\hat{X}^{(j)}$ is the jth column of $\hat{X}$. Let $\hat{Y}$ denote the right hand side of Equation (20).

$$\begin{aligned}(\hat{Y}\hat{X})_{jj} &= \hat{X}_{jj}\hat{Y}_{jj} + \hat{X}_{j-1j}\hat{Y}_{j-1j} + \hat{X}_{jj+1}\hat{Y}_{jj+1} \\ &= \hat{X}_{jj}H_{jj} + \hat{X}_{j-1j}H_{j-1j} + \hat{X}_{jj+1}H_{jj+1} \\ &= 1\end{aligned}$$

The third equality is by using the following alternative form of Equation (12):

$$(\hat{X}^{(1)})_{i,j} = \begin{cases} 0 & \text{if } j-i>1 \\ \frac{-H_{i,i+1}}{(H_{ii}H_{i+1,i+1}-H^2_{i+1,i+1})} & \text{if } j=i+1 \\ \frac{1}{H_{ii}}\left(1+\sum_{j\in \text{neig}_{\mathcal{G}}(i)} \frac{H^2_{ij}}{H_{ii}H_{jj}-H^2_{ij}}\right) & \text{if } i=j \end{cases}, \tag{21}$$

where $i<j$. Similarly, the offdiagonals of $\hat{Y}\hat{X}$ can be evaluated to be zero as follows.

$$\begin{aligned}(\hat{Y}\hat{X})_{ij} &= \hat{Y}_{ij}\hat{X_j}j + \hat{Y}_{ij-1}\hat{X}_{j-1j} + \hat{Y}_{ij+1}\hat{X}_{j+1j} \\ &= \hat{Y}_{ij}\hat{X}_{jj} + \hat{Y}_{ij}\frac{H_{j-1j-1}}{H_{j-1j}} + \hat{Y}_{ij}\frac{H_{jj+1}}{H_{jj}}\hat{X}_{j+1j} \\ &= 0\end{aligned}$$

□

Lemma 4. *Let* $y \in \mathbb{R}^n$, $\beta = \max_t \max_{i\in[n-1]}(H_t)_{ii+1}/\sqrt{(H_t)_{ii}(H_t)_{i+1i+1}} < 1$, $C = \sum_{i=1}^n y_i^2 (G^{(i)}_\infty)^2$ *, where* $g^{(i)}_{1:T} = [(g_1)_i, \ldots (g_T)_i]$ *and* $G^{(i)}_\infty = \|g^{(i)}_{1:T}\|_\infty$ *, then*

$$y^T X_t^{-1} y \leq (Ct + \epsilon \|y\|_2^2)\left(\frac{1+\beta}{1-\beta}\right)$$

Proof. Let $\tilde{X}_t^{-1} = \text{diag}(H_t)^{-1/2}\hat{X}_t\,\text{diag}(H_t)^{-1/2}$

$$y^T X_t^{-1} y \leq \left\|\text{diag}(H_t)^{1/2} y\right\|_2^2 \left\|\tilde{X}_t^{-1}\right\|_2 \tag{22}$$

Using the identity of spectral radius $\rho(X) \leq \|X\|_\infty$ and since $\tilde{X}$ is positive definite, $\left\|\tilde{X}_t^{-1}\right\|_2 \leq \|\tilde{X}_t^{-1}\|_\infty$

$$\begin{aligned}\left\|\tilde{X}_t^{-1}\right\|_2 &\leq \max_i \left\{\sum_j \left|(\tilde{X}_t^{-1})_{ij}\right|\right\} \\ &\leq 1 + 2(\beta + \beta^2 + \ldots) \\ &\leq \frac{1+\beta}{1-\beta}\end{aligned}$$

The second inequality is using Lemma 3. Using $(H_t)_{ii} = \left\|g^{(i)}_{1:T}\right\|_2^2 + \epsilon$ in Equation (22) will give the lemma. □

Lemma 5 (Upperbound of T_1)**.** $T_1 \leq \frac{(C+\epsilon D_2^2)}{2\eta} \cdot \frac{1+\beta}{1-\beta}$*, where* $D_2 = \max_{t\in[T]} \|w_t - w^*\|_2$*, where* $C = \max_t(\sum_{i=1}^n (y_i^{(t)})^2 (G^{(i)}_\infty)^2)$.

Proof. Since X_T is positive definite

$$\begin{aligned}T_1 &\leq \frac{\|w_1 - w^*\|^2_{X_1^{-1}}}{2\eta} \\ &\leq \frac{(y^{(1)})^T X_1^{-1} y^{(1)}}{2\eta}\end{aligned}$$

Using Lemma 4 proves the lemma. □

Upperbounding T_2
Let $y_t = w_t - w^*$, $P_t = y_t^T X_t^{-1} y_t$, and $Q_t = y_t^T X_{t-1}^{-1} y_t$. Let $P = [P_2, \cdots, P_T]$ and $Q = [Q_2, \cdots, Q_T]$ be the two array representations. Then,

$$\begin{aligned}
2\eta T_2 &= \sum_{i=1}^{T-1} y_{t+1}^T \left(X_{t+1}^{-1} - X_t^{-1}\right) y_{t+1} \\
&= \sum_{i=2}^{T} y_t^T \left(X_t^{-1} - X_{t-1}^{-1}\right) y_t \\
&= \sum_{i=2}^{T} y_t^T X_t^{-1} y_t - y_t^T X_{t-1}^{-1} y_t \\
&= \sum_{i=2}^{T} P_t - Q_t \\
&= \|P - Q\|_1
\end{aligned} \tag{23}$$

In order to upper bound this, we derive here a generalized version of Pinsker's inequality for our setting.

Lemma 6. *Let $A = [A_1, \cdots, A_T]$ and $B = [B_1, \cdots, B_T]$ be two T-length arrays. Let $B_i, A_i > 0 \; \forall i \in [1, T+1]$. Then,*

$$\|A - B\|_1^2 \leq \frac{2}{3} \left(\sum_{i=1}^{T} (2A_i + B_i) \right) D_{KL}(B, A)$$

where the generalized KL-Divergence $D_{KL}(B, A) = \sum_{i=1}^{T} \left(B_i log\left(\frac{B_i}{A_i}\right) - B_i + A_i \right)$.

Proof. This proof is a mock up of Pinsker's inequality for probability spaces. First step involves using the following identity, for $t \geq -1$

$$(1+t)log(1+t) - t \geq \frac{1}{2} \cdot \frac{t^2}{1 + \frac{t}{3}}$$

Let $t_i = \frac{B_i}{A_i} - 1$. Then, $-1 \leq t_i$. Therefore,

$$\begin{aligned}
D_{KL}(B, A) &= \sum_{i=1}^{T} \left(B_i \log\left(\frac{B_i}{A_i}\right) - B_i + A_i \right) \\
&= \sum_{i=1}^{T} A_i \left(\frac{B_i}{A_i} \log\left(\frac{B_i}{A_i}\right) - \frac{B_i}{A_i} + 1 \right) \\
&= \sum_{i=1}^{T} A_i \left((t_i + 1) \log(t_i + 1) - t_i \right) \\
&\geq \sum_{i=1}^{T} \frac{A_i t_i^2}{2\left(1 + \frac{t_i}{3}\right)}
\end{aligned}$$

$$
\begin{aligned}
D_{KL}(B, A) &= \sum_{i=1}^{T} \frac{A_i t_i^2}{2\left(1+\frac{t_i}{3}\right)} \cdot \frac{\sum_{j=1}^{T} A_j(1+\frac{t_j}{3})}{\sum_{k=1}^{T} A_k(1+\frac{t_k}{3})} \\
&\geq \frac{\left(\sum_{i=1}^{T} A_i |t_i|\right)^2}{2\sum_{k=1}^{T} A_k(1+\frac{t_k}{3})} && \text{(Using Cauchy-Schwartz)} \\
&= \frac{\left(\sum_{i=1}^{T} |A_i - B_i|\right)^2}{\frac{2}{3}\sum_{k=1}^{T}(2A_k + B_k)} \\
&= \frac{\|A-B\|_1^2}{\frac{2}{3}\sum_{j=1}^{T}(2A_i + B_i)}
\end{aligned}
$$

□

Using the above to bound Equation (23) with $B := P$ and $A := Q$, we get

$$
\begin{aligned}
2\eta T_2 = \|P - Q\|_1 &\leq \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) D_{KL}(P, Q)} \\
&= \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \sum_{t=2}^{T} D_{KL}(P_i, Q_i)} \\
&= \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \sum_{t=2}^{T} D_{KL}(y_t^T X_t^{-1} y_t, y_t^T X_{t-1}^{-1} y_t)} \\
&= \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \sum_{t=2}^{T} D_{KL}(\bar{y_t}^T \bar{X}_t^{-1} \bar{y}_t, \bar{y_t}^T \bar{X}_{t-1}^{-1} \bar{y}_t)} \\
&\leq \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \cdot \sum_{t=2}^{T} \|\bar{y}_t\|_2^2 \cdot D_{vN}(\bar{X}_t^{-1}, \bar{X}_{t-1}^{-1})} \qquad (24)
\end{aligned}
$$

Here, $\bar{y}_t = \Lambda_t y_t$, $\bar{X}_t = \Lambda_t X_t \Lambda_t$, $\bar{X}_{t-1} = \Lambda_t X_{t-1} \Lambda_t$ D_{vN} is von-Neuman Divergence, and last inequality is from Lindblad (1975). We upper bound Equation (24) using the following lemma.

Lemma 7. *Let A, B be two PD matrices. Then,*

$$
D_{vN}(A, B) \leq \lambda_{max}(A) \cdot \mathrm{D}_{\ell\mathrm{d}}(B, A)
$$

Proof. Let $A = V \Lambda V^T$ and $B = U \Theta U^T$

$$
\begin{aligned}
D_{vN}(A, B) &= \sum_{i,j} (v_i^T u_j) \left(\lambda_i \log\left(\frac{\lambda_i}{\theta_j}\right) - \lambda_i + \theta_j\right) \\
&= \sum_{i,j} (v_i^T u_j) \lambda_i \left(\log\left(\frac{\lambda_i}{\theta_j}\right) - 1 + \frac{\theta_j}{\lambda_i}\right) \\
&\leq \max_i \lambda_i \sum_{i,j} (v_i^T u_j) \left(-\log\left(\frac{\theta_j}{\lambda_i}\right) - 1 + \frac{\theta_j}{\lambda_i}\right) \\
&= \lambda_{max}(A) \cdot \mathrm{D}_{\ell\mathrm{d}}(B, A)
\end{aligned}
$$

□

Combining the above result, we get

$$
\begin{aligned}
T_2 &= \frac{1}{2\eta} \cdot \|P - Q\|_1 \\
&\leq \frac{1}{2\eta} \cdot \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \cdot \sum_{t=2}^{T} \|\bar{y}_t\|_2^2 \left\|\bar{X}_t^{-1}\right\|_2 \cdot \mathrm{D}_{\ell\mathrm{d}}(\bar{X}_{t-1}^{-1}, \bar{X}_t^{-1})} \\
&\leq \frac{1}{2\eta} \cdot \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \cdot \sum_{t=2}^{T} \|\bar{y}_t\|_2^2 \left\|\bar{X}_t^{-1}\right\|_2 \cdot \mathrm{D}_{\ell\mathrm{d}}(\bar{X}_{t-1}^{-1}, \bar{X}_t^{-1})}
\end{aligned}
$$

The second inequality is due to scale invariance of LogDet divergence. If we set $\Lambda_t = \mathrm{diag}(H_t)^{1/2}$, then using Lemma 4

$$
\begin{aligned}
T_2 &\leq \frac{1}{2\eta} \cdot \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \cdot \sum_{t=2}^{T}(Ct + \epsilon \|y_t\|_2^2)\left(\frac{1+\beta}{1-\beta}\right) \cdot \mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1})} \\
&\leq \frac{1}{2\eta} \cdot \sqrt{\frac{2}{3}\left(\sum_{t=2}^{T}(2Q_t + P_t)\right) \cdot (CT + \epsilon D_2^2)\left(\frac{1+\beta}{1-\beta}\right) \cdot \sum_{t=2}^{T} \mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1})} \\
&\leq \frac{1}{2\eta} \cdot (CT + \epsilon D_2^2)\left(\frac{1+\beta}{1-\beta}\right) \cdot \sqrt{\frac{2}{3} \cdot T \cdot \sum_{t=2}^{T} \mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1})} \qquad (25)
\end{aligned}
$$

where, $D_2 = \max_t \|y_t\|_2$, the second and third inequality is using Lemma 4. Now to bound the $\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1})$ term in the above equation, we develop the following lemma.

Lemma 8. $\mathrm{Tr}(X_t H_t) = n$, $\forall t \in \{2, \ldots, T\}$ *in Algorithm 1 with any* $\mathcal{G}$

Proof. $P_{\mathcal{G}}(X_t^{-1}) = H_t$ is an optimality condition of LogDet subproblem Equation (11)

$$
\begin{aligned}
\mathrm{Tr}(X_t H_t) &= \mathrm{Tr}(X_t P_{\mathcal{G}}(X_t^{-1})) \\
&= \mathrm{Tr}(X_t X_t^{-1}) \\
&= n
\end{aligned}
$$

The second equality is because X_t follows the sparsity graph $\mathcal{G}$. □

Lemma 9.

$$
\begin{aligned}
\sum_{t=2}^{T} \mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1}) &\leq \log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_1^{-1}\right)}\right) \\
&\leq \sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right)
\end{aligned}
$$

where $\beta_i = \frac{\left\langle g_{1:T}^{(i)}, g_{1:T}^{(i+1)} \right\rangle}{\sqrt{(\epsilon + \left\|g_{1:T}^{(i)}\right\|_2^2) \cdot (\epsilon + \left\|g_{1:T}^{(i+1)}\right\|_2^2)}}$

Proof. LogDet divergence is defined as follows

$$
\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1}) = -\log\left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)}\right) + \mathrm{Tr}(X_{t-1}^{-1} X_t) - n
$$

The second two terms on right hand side can be simplified analyzed as follows.

$$\begin{aligned}\mathrm{Tr}(X_{t-1}^{-1}X_t) - n &= \mathrm{Tr}(X_{t-1}^{-1}X_t - X_t^{-1}X_t) \\ &= \mathrm{Tr}((X_{t-1}^{-1} - X_t^{-1})X_t) \\ &= -g_t^T X_t g_t \\ &< 0 \end{aligned} \tag{26}$$

Thus

$$\begin{aligned}\sum_{t=2}^{T} \mathrm{D}_{\ell\mathrm{d}}\left(X_{t-1}^{-1}, X_t^{-1}\right) &\leq \sum_{t=2}^{T} -\log\left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)}\right) \\ &\leq \log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_1^{-1}\right)}\right)\end{aligned}$$

Since LogDet divergence is positive

$$\begin{aligned}\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1}) &= -\log\left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)}\right) + \mathrm{Tr}(X_{t-1}^{-1}X_t) - n \\ &\geq 0 \\ \implies \log\left(\frac{\det\left(X_t^{-1}\right)}{\det\left(X_{t-1}^{-1}\right)}\right) &\geq 0\end{aligned} \tag{27}$$

The second inequality uses Equation (26). Now, using Equation (13), which is in cholesky decomposition format,

$$\begin{aligned}\log(\det\left(X_T^{-1}\right)) &= \sum_{i=1}^{n-1} \log\left((H_T)_{ii} - (H_T)_{ii+1}^2/(H_T)_{i+1i+1}\right) + \log((H_T)_{nn}) \\ &\leq \sum_{i=1}^{n-1} \log(1-\beta_i^2) + \sum_{i=1}^{n} \log((H_T)_{ii})\end{aligned}$$

Using the above, we can expand the log deteriminatn difference.

$$\begin{aligned}\log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_1^{-1}\right)}\right) &\leq \log\left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_0^{-1}\right)}\right) \\ &\leq \sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right)\end{aligned} \tag{28}$$

Let $X_0 = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, H_0^{-1})$, the above inequality is since $\log\left(\frac{\det\left(X_1^{-1}\right)}{\det\left(X_0^{-1}\right)}\right) \geq 0$ using Equation (27) □

Lemma 10 (Upperbound of T_2)**.**

$$T_2 \leq \frac{1}{2\eta} \cdot (CT + \epsilon D_2^2)\left(\frac{1+\beta}{1-\beta}\right) \cdot \sqrt{\frac{2}{3} \cdot T \cdot \left(\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right)\right)}$$

Proof. The proof directly follows from Equation (25) and Lemma 9 □

Lemma 11 (Upperbound of T_3)**.** *Let* $X_0 = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, H_0^{-1})$

$$\begin{aligned} T_3 &= \sum_{t=1}^{T} \frac{\eta}{2} \cdot g_t^T X_t g_t \\ &\leq \frac{\eta}{2} \cdot \log \left(\frac{\det\left(X_T^{-1}\right)}{\det\left(X_0^{-1}\right)} \right) \\ &\leq \frac{\eta}{2} \cdot \sum_{i=1}^{n} \log \left(1 + \frac{\left\| g_{1:T}^{(i)} \right\|_2^2}{\epsilon} \right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right) \end{aligned}$$

where $\beta_i = \frac{((g_{1:T}^{(i)})^T g_{1:T}^{(i+1)})^2}{(\epsilon + \left\| g_{1:T}^{(i)} \right\|_2^2) \cdot (\epsilon + \left\| g_{1:T}^{(i+1)} \right\|_2^2)}$

Proof.

$$\begin{aligned} g_t^T X_t g_t &= \mathrm{Tr}(X_t g_t g_t^T) \\ &= \mathrm{Tr}(X_t (X_t^{-1} - X_{t-1}^{-1})) \\ &= n - \mathrm{Tr}(X_t X_{t-1}^{-1}) \\ &\leq -\log \left(\frac{\det\left(X_{t-1}^{-1}\right)}{\det\left(X_t^{-1}\right)} \right) \end{aligned}$$

The second equality is using Lemma 3 and that X_t's sparsity graph is tridiagonal. The first inequality is using the property $\mathrm{D}_{\ell\mathrm{d}}(X_{t-1}^{-1}, X_t^{-1}) \geq 0$ of LogDet divergence. Thus summing up and using Equation (28) will give the lemma. □

Putting together T_1, T_2 and T_3 from Lemma 5, Lemma 10 and Lemma 11

$$T_1 \leq \frac{(C + \epsilon D_2^2)}{2\eta} \cdot \frac{1+\beta}{1-\beta},$$

$$T_2 \leq \frac{1}{2\eta}(CT + \epsilon D_2^2) \left(\frac{1+\beta}{1-\beta} \right) \sqrt{\frac{2}{3} T \left(\sum_{i=1}^{n} \log \left(1 + \frac{\left\| g_{1:T}^{(i)} \right\|_2^2}{\epsilon} \right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right) \right)}, \tag{29}$$

$$T_3 \leq \frac{\eta}{2} \cdot \sum_{i=1}^{n} \log \left(1 + \frac{\left\| g_{1:T}^{(i)} \right\|_2^2}{\epsilon} \right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right) \tag{30}$$

If we set $\eta = \frac{C^{1/2} T^{3/4} \cdot (\frac{1+\beta}{1-\beta})^{1/2}}{\left(\sum_{i=1}^{n} \log \left(1 + \frac{\left\| g_{1:T}^{(i)} \right\|_2^2}{\epsilon} \right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right) \right)^{1/4}}$, then

$$\begin{aligned} R_T &\leq T_1 + T_2 + T_3 \\ &\leq \mathcal{O}\left(C^{1/2} \cdot T^{3/4} \cdot \left(\frac{1+\beta}{1-\beta} \right)^{1/2} \cdot \left(\sum_{i=1}^{n} \log \left(1 + \frac{\left\| g_{1:T}^{(i)} \right\|_2^2}{\epsilon} \right) + \sum_{i=1}^{n-1} \log\left(1 - \beta_i^2\right) \right)^{3/4} \right) \end{aligned}$$

A.3 $\mathcal{O}(T^{1/2})$ Regret upper bound

In this section we derive a regret upper bound with a $\mathcal{O}(T^{1/2})$ growth, in contrast to the $\mathcal{O}(T^{3/4})$ obtained in A.2. In (29), $T_2 = \sum_{t=2}^{T} (w_t - w^*)^T (X_t^{-1} - X_{t-1}^{-1})(w_t - w^*)$ is of the order $\mathcal{O}(T^{3/2})$, which can be reduced to $\mathcal{O}(T)$, by upper bounding each entry of $X_t^{-1} - X_{t-1}^{-1}$ individually. The following lemma helps in constructing a telescoping argument to bound $\left|(X_t^{-1} - X_{t-1}^{-1})_{i,j}\right|$.

Lemma 12. *Let $H, \tilde{H} \in S_n^{++}$, such that $\tilde{H} = H + gg^T$, where $g \in \mathbb{R}^n$, then*

$$\frac{\tilde{H}_{ij}}{\sqrt{\tilde{H}_{ii}\tilde{H}_{jj}}} - \frac{H_{ij}}{\sqrt{H_{ii}H_{jj}}} = \frac{g_i g_j}{\sqrt{\tilde{H}_{ii}\tilde{H}_{jj}}} + \frac{H_{ij}}{\sqrt{H_{ii}H_{jj}}}\left(\sqrt{\frac{H_{ii}H_{jj}}{\tilde{H}_{ii}\tilde{H}_{jj}}} - 1\right) = \theta_{ij}$$

Proof.

$$\begin{aligned}\frac{\tilde{H}_{ij}}{\sqrt{\tilde{H}_{ii}\tilde{H}_{jj}}} - \frac{H_{ij}}{\sqrt{H_{ii}H_{jj}}} &= \frac{1}{\sqrt{H_{ii}H_{jj}}}(\tilde{H}_{ij}\frac{\sqrt{H_{ii}H_{jj}}}{\sqrt{\tilde{H}_{ii}\tilde{H}_{jj}}} - H_{ij}) \\ &= \frac{1}{\sqrt{H_{ii}H_{jj}}}\left(g_i g_j \frac{\sqrt{H_{ii}H_{jj}}}{\sqrt{\tilde{H}_{ii}\tilde{H}_{jj}}} + H_{ij}\left(\frac{\sqrt{H_{ii}H_{jj}}}{\sqrt{\tilde{H}_{ii}\tilde{H}_{jj}}} - 1\right)\right)\end{aligned}$$

□

Lemma 13. *Let $H, \tilde{H} \in S_n^{++}$, such that $\tilde{H} = H + gg^T$, where $g \in \mathbb{R}^n$. Also, $\tilde{Y} = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, \tilde{H})$ and $Y = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}(X, H)$, where $\mathcal{G}$ is a chain graph, then*

$$\left|(\tilde{Y}^{-1} - Y^{-1})_{ii+k}\right| \leq G_\infty^2 \kappa(k\beta + k + 2)\beta^{k-1},$$

where $i, i+k \leq n$, $G_\infty = \|g\|_\infty$ and $\max_{i,j} |H_{ij}|/\sqrt{H_{ii}H_{jj}} \leq \beta < 1$. Let $\kappa(\mathrm{diag}(H)) :=$ condition number of the diagonal part of H, then $\kappa := \max(\kappa(\mathrm{diag}(H)), \kappa(\mathrm{diag}(\tilde{H})))$.

Proof. Using Lemma 3 will give the following:

$$\begin{aligned}\left|(\tilde{Y}^{-1} - Y^{-1})_{ii+k}\right| &= \left|\frac{\tilde{H}_{ii+1}\dots\tilde{H}_{i+k-1i+k}}{\tilde{H}_{i+1i+1}\dots\tilde{H}_{i+ki+k}} - \frac{H_{ii+1}\dots H_{i+k-1i+k}}{H_{i+1i+1}\dots H_{i+ki+k}}\right| \\ &= \left|\sqrt{\tilde{H}_{ii}}\tilde{N}_{ii+1}\dots\tilde{N}_{i+k-1i+k}\sqrt{\tilde{H}_{i+ki+k}} - \sqrt{H_{ii}}N_{ii+1}\dots N_{i+k-1i+k}\sqrt{H_{i+ki+k}}\right| \\ &= \sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}\left|\tilde{N}_{ii+1}\dots\tilde{N}_{i+k-1i+k} - N_{ii+1}\dots N_{i+k-1i+k}\sqrt{\frac{H_{ii}H_{i+ki+k}}{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}}\right|\end{aligned}$$

□

where $N_{ij} = H_{ij}/\sqrt{H_{ii}H_{jj}}$. Expanding $\tilde{N}_{ii+1} = N_{ii+1} + \theta_{ii+1}$ (from Lemma 12), subsequently $\tilde{N}_{ii+2} = N_{ii+2} + \theta_{ii+2}$ and so on will give

$$\begin{aligned}&\left|\tilde{N}_{ii+1}\dots\tilde{N}_{i+k-1i+k} - N_{ii+1}\dots N_{i+k-1i+k}\sqrt{\frac{H_{ii}H_{i+ki+k}}{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}}\right| = \\ &\left|\theta_{ii+1}\tilde{N}_{i+1i+2}\dots\tilde{N}_{i+k-1i+k} + N_{ii+1}\left(\tilde{N}_{i+1i+2}\dots\tilde{N}_{i+k-1i+k} - N_{i+1i+2}\dots N_{i+k-1i+k}\sqrt{\frac{H_{ii}H_{i+ki+k}}{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}}\right)\right| \\ &= |\theta_{ii+1}\tilde{N}_{i+1i+2}\dots\tilde{N}_{i+k-1i+k} + N_{ii+1}\theta_{i+1i+2}\tilde{N}_{ii+3}\dots\tilde{N}_{i+k-1i+k} + \dots + N_{ii+1}\dots N_{ii+k-1}\theta_{i+k-1i+k} \\ &+ N_{ii+1}\dots N_{ii+k}\left(1 - \sqrt{\frac{H_{ii}H_{i+ki+k}}{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}}\right)| \\ &\leq (\sum_{l=0}^{k-1}|\theta_{i+li+l+1}|)\beta^{k-1} + \beta^{k-1}\left|1 - \sqrt{\frac{H_{ii}H_{i+ki+k}}{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}}\right|, \\ &\implies \left|(\tilde{Y}^{-1} - Y^{-1})_{ii+k}\right| \leq \sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}\left((\sum_{l=0}^{k-1}|\theta_{i+li+l+1}|)\beta^{k-1} + \beta^{k-1}\left|1 - \sqrt{\frac{H_{ii}H_{i+ki+k}}{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}}\right|\right)\end{aligned}$$

where $\max_{i,j}|N_{i,j}|,\ \max_{i,j}|\tilde{N}_{i,j}| \leq \beta$. Expanding $\theta_{i+li+l+1}$ from Lemma 12 in the term $|\theta_{i+li+l+1}|\sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}$ will give:

$$
\begin{aligned}
|\theta_{i+li+l+1}|\sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}} &= \left|\sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}\frac{g_{i+l}g_{i+l+1}}{\sqrt{\tilde{H}_{i+li+l}\tilde{H}_{i+l+1i+l+1}}} + \right. \\
&\left. \sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}N_{i+li+l+1}\left(\sqrt{\frac{H_{i+li+l}H_{i+l+1i+l+1}}{\tilde{H}_{i+li+l}\tilde{H}_{i+l+1i+l+1}}}-1\right)\right| \\
&\leq \left|\sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}\frac{g_{i+l}g_{i+l+1}}{\sqrt{\tilde{H}_{i+li+l}\tilde{H}_{i+l+1i+l+1}}}\right| + \\
&\left|\sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}N_{i+li+l+1}\left(1-\sqrt{\frac{H_{i+li+l}H_{i+l+1i+l+1}}{\tilde{H}_{i+li+l}\tilde{H}_{i+l+1i+l+1}}}\right)\right|
\end{aligned}
$$

Since $H_{i+li+l}H_{i+l+1i+l+1} \leq \tilde{H}_{i+li+l}\tilde{H}_{i+l+1i+l+1}$,

$$
\begin{aligned}
1-\sqrt{\frac{H_{i+li+l}H_{i+l+1i+l+1}}{\tilde{H}_{i+li+l}\tilde{H}_{i+l+1i+l+1}}} &\leq \max\left(1-\frac{H_{i+li+l}}{\tilde{H}_{i+li+l}}, 1-\frac{H_{i+l+1i+l+1}}{\tilde{H}_{i+l+1i+l+1}}\right) \\
&\leq \max\left(\frac{g_{i+l}^2}{\tilde{H_{i+li+l}}}, \frac{g_{i+l+1}^2}{\tilde{H_{i+l+1i+l+1}}}\right)
\end{aligned}
$$

Using the above, $H_{i,i}/H_{j,j} \leq \kappa$, and $|g_i| \leq G_\infty, \forall i,j \in [n]$, gives

$$
\begin{aligned}
\sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}|\theta_{i+li+l+1}| &\leq G_\infty^2\kappa + \beta G_\infty^2\kappa \\
&\leq G_\infty^2\kappa(1+\beta)
\end{aligned}
$$

Thus the following part of $\left|\left(\tilde{Y}^{-1}-Y^{-1}\right)_{ii+k}\right|$ can be upperbounded:

$$
\sqrt{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}\left((\sum_{l=0}^{k-1}|\theta_{i+li+l+1}|)\beta^{k-1}\right) \leq G_\infty^2\kappa(1+\beta)k\beta^{k-1}
$$

Also, $\beta^{k-1}\left|1-\sqrt{\frac{H_{ii}H_{i+ki+k}}{\tilde{H}_{ii}\tilde{H}_{i+ki+k}}}\right| \leq \beta^{k-1}\kappa G_\infty^2$, so

$$
\left|\left(\tilde{Y}^{-1}-Y^{-1}\right)_{ii+k}\right| \leq G_\infty^2\kappa(k\beta+k+2)\beta^{k-1}
$$

Lemma 14 ($\mathcal{O}(T)$ upper bound of T_2)**.** *Given that $\kappa(\mathrm{diag}(H_t)) \leq \kappa$, $\|w_t - w^*\|_2 \leq D_2$, $\max_{i,j}|(H_t)_{ij}|/\sqrt{(H_t)_{ii}(H_t)_{jj}} \leq \beta < 1, \forall t \in [T]$ in Algorithm 1, then T_2 in (29) can be bounded as follows:*

$$
T_2 \leq \mathcal{O}\left(\frac{T}{2\eta(1-\beta)^2}(G_\infty D_2)^2\kappa\right)
$$

Proof. Note that $T_2 = \frac{1}{2\eta}\cdot\sum_{t=1}^{T-1}(w_{t+1}-w^*)^T(X_{t+1}^{-1}-X_t^{-1})(w_{t+1}-w^*) \leq \sum_{t=1}^{T-1}D_2^2\left\|(X_{t+1}^{-1}-X_t^{-1})\right\|_2$. Using $\|A\|_2 = \rho(A) \leq \|A\|_\infty$ for symmetric matrices A, we get

$$
\begin{aligned}
\left\|X_{t+1}^{-1}-X_t^{-1}\right\|_2 &\leq \|X_{t+1}^{-1}-X_t^{-1}\|_\infty \\
&= \max_i(\sum_j\left|(X_{t+1}^{-1}-X_t^{-1})_{ij}\right|) \\
&\leq \mathcal{O}(\frac{G_\infty^2\kappa}{(1-\beta)^2})
\end{aligned}
$$

The third inequality is using Lemma 13. Expanding T_2 with this bound gives the result. □

Theorem 6 ($\mathcal{O}(T^{1/2})$ regret upper bound)**.** *Setting*

$$\eta = \frac{T^{1/2} D_2 G_\infty \sqrt{\kappa}}{(1-\beta)\sqrt{\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right)}},$$

gives the following regret

$$R_T \leq \mathcal{O}\left(\frac{1}{(1-\beta)} T^{1/2} D_2 G_\infty \sqrt{\kappa} \sqrt{\sum_{i=1}^{n} \log\left(1 + \frac{\left\|g_{1:T}^{(i)}\right\|_2^2}{\epsilon}\right) + \sum_{i=1}^{n-1} \log\left(1-\beta_i^2\right)}\right),$$

where condition number $\kappa(\mathrm{diag}(H_t)) \leq \kappa$, $\|w_t - w^*\|_2 \leq D_2$, $\max_{i,j} |(H_t)_{ij}| / \sqrt{(H_t)_{ii}(H_t)_{jj}} \leq \beta < 1$, $\forall t \in [T]$ *and* g_t, H_t *are as defined in Algorithm 1.*

Proof. The result follows from using (30) and Lemma 14. □

A.4 Numerical stability

Theorem 7 (*Full version of Theorem 4*)**.** *Let* $H \in S_n^{++}$ *such that* $H_{ii} = 1$, *for* $i \in [n]$ *and a symmetric perturbation* ΔH *such that* $\Delta H_{ii} = 0$, *for* $i \in [n]$ *and* $H + \Delta H \succ 0$. *Let* $\hat{X} = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}\left(X, H^{-1}\right)$ *and* $\hat{X} + \Delta\hat{X} = \arg\min_{X \in S_n(\mathcal{G})^{++}} \mathrm{D}_{\ell\mathrm{d}}\left(X, (H+\Delta H)^{-1}\right)$, *here* $\mathcal{G} :=$ *chain/tridiagonal sparsity graph and* $S_n(\mathcal{G})^{++}$ *denotes positive definite matrices which follows the sparsity pattern* $\mathcal{G}$.

$$\begin{aligned}\kappa_{\ell d} &= \max_{|i-j|\leq 1} \limsup_{\epsilon \to 0} \left\{ \frac{\left|\Delta\hat{X}_{ij}\right|}{\epsilon\left|\hat{X}_{ij}\right|} : |\Delta H_{k,l}| \leq |\epsilon H_{k,l}|, (k,l) \in E_{\mathcal{G}} \right\} \\ &\leq \max_{i \in [n-1]} 1/(1-\beta_i^2)\end{aligned}$$

where, $\kappa_{\ell d} :=$ *condition number of the LogDet subproblem,* $\kappa_2(.) :=$ *condition number of a matrix in* ℓ_2 *norm,* $\beta_i = H_{ii+1}/\sqrt{H_{ii}H_{i+1i+1}}$

Proof. Consider the offdiagonals for which $(\hat{X} + \Delta\hat{X})_{ii+1} = -H_{ii+1}/(1 - H_{ii+1}^2) = f(H_{ii+1})$,where $f(x) = -x/(1-x^2)$. Let $y = f(x)$, $\hat{y} = f(x + \Delta x)$ and $|\Delta x / x| \leq \epsilon$ then using Taylor series

$$\begin{aligned}\left|\frac{(\hat{y}-y)}{y}\right| &= \left|\frac{x f'(x)}{f(x)}\right| \left|\frac{\Delta x}{x}\right| + O((\Delta x)^2) \\ \implies \lim_{\epsilon \to 0} \left|\frac{(\hat{y}-y)}{\epsilon y}\right| &\leq \frac{x f'(x)}{f(x)}\end{aligned}$$

Using the above inequality, with $x := H_{ii+1}$ and $y := \hat{X}_{ii+1}$,

$$\begin{aligned}\lim_{\epsilon \to 0} \left|\frac{\Delta\hat{X}_{ii+1}}{\epsilon\hat{X}_{ii+1}}\right| &\leq \frac{1 + H_{ii+1}^2}{1 - H_{ii+1}^2} \\ &\leq \frac{2}{1 - H_{ii+1}^2}\end{aligned} \tag{31}$$

Let $g(x) = x^2/(1-x^2)$, let $y_1 = g(w_1)$, $y_2 = g(x_2)$, $\hat{y}_1 = g(w_1 + \Delta x)$, $\hat{y}_2 = g(x_2 + \Delta x)$. Using Taylor series

$$\begin{aligned}\left|\frac{(\hat{y}_1 - y_1)}{y_1}\right| &= \left|\frac{x_1 f'(x_1)}{f(x_1)}\right| \left|\frac{\Delta x_1}{x_1}\right| + O((\Delta x_1)^2) \\ \left|\frac{(\hat{y}_2 - y_2)}{y_2}\right| &= \left|\frac{x_2 f'(x_2)}{f(x_2)}\right| \left|\frac{\Delta x_2}{x_2}\right| + O((\Delta x_2)^2) \\ \implies \lim_{\epsilon \to 0} \frac{\Delta y_1 + \Delta y_2}{\epsilon(1 + y_1 + y_2)} &\leq \max\left(\frac{2}{1-x_1^2}, \frac{2}{1-x_2^2}\right)\end{aligned}$$

Table 3: **float32 experiments on Autoencoder benchmark using different band sizes.** Band size 0 corresponds to diag-SONew and 1 corresponds to tridiag-SONew. We see the training loss getting better as we increase band size

Band size	0 (diag-SONew)	1 (tridiag-SONew)	4	10
Train CE loss	53.025	51.723	51.357	51.226

Putting $x_1 \coloneqq H_{ii+1}$, $x_2 \coloneqq H_{ii-1}$ and analyzing $y_1 \coloneqq H_{ii+1}^2/(1 - H_{ii+1}^2)$ and $y_2 \coloneqq H_{ii-1}^2/(1 - H_{ii-1}^2)$ will result in the following

$$\lim_{\epsilon \to 0} \left| \frac{\Delta \hat{X}_{ii}}{\hat{X}_{ii}} \right| \leq \max \left(\frac{2}{1 - H_{ii+1}^2}, \frac{2}{1 - H_{ii-1}^2} \right) \tag{32}$$

Since $\hat{X}_{ii} = 1 + H_{ii+1}^2/(1 - H_{ii+1}^2) + H_{ii-1}^2/(1 - H_{ii-1}^2)$. Putting together Equation (32) and Equation (31), the theorem is proved. □

Proof of Lemma 1

For $b = 1$, if $g_{1:T}^{(j)} = g_{1:T}^{(j+1)}$, then $H_{jj+1} = H_{jj} = H_{j+1j+1} = \left\| g_{1:T}^{(j)} \right\|_2^2$, thus $H_{jj} - H_{jj+1}^2/H_{j+1j+1} = 0$.
For $b > 1$, since $H_{I_j I_j}$, using Guttman rank additivity formula, $\text{rank}(H_{jj} - H_{jj+1}^2/H_{j+1j+1}) = \text{rank}(H_{J_j J_j}) - rank(H_{I_j I_j}) = 0$, thus $H_{jj} - H_{jj+1}^2/H_{j+1j+1} = 0$.
Furthermore, if $\text{rank}(H) \leq b$, then all $b + 1 \times b + 1$ principal submatrices of H have rank b, thus $\forall j$, $H_{J_j J_j}$ have a rank b, thus D_{jj} for all j are undefined.

Proof of Theorem 5

Let $I_i = \{j : i < j, (i,j) \in E_{\mathcal{G}}\}$ and $I_i' = \left\{j : i < j, (i,j) \in E_{\tilde{\mathcal{G}}}\right\}$ Let $K = \left\{i : H_{ii} - H_{I_i i}^T H_{I_i I_i}^{-1} H_{I_i i} \text{ is undefined or } 0, i \in [n]\right\}$ denote vertices which are getting removed by the algorithm, then for the new graph $\tilde{\mathcal{G}}$, $D_{ii} = 1/H_{ii}, \forall i \in K$ since $H_{ii} > 0$.
Let $\bar{K} = \left\{i : H_{ii} - H_{I_i i}^T H_{I_i I_i}^{-1} H_{I_i i} > 0, i \in [n]\right\}$. Let for some $j \in \bar{K}$, if

$$l = \arg\min \left\{i : j < i, i \in K \cap I_j\right\},$$

denotes the nearest connected vertex higher than j for which D_{ll} is undefined or zero, then according to the definition $E_{\tilde{\mathcal{G}}}$ in Algorithm 3, $I_j' = \{j+1, \dots l-1\} \subset I_j$, since D_{jj} is well-defined, $H_{I_j I_j}$ is invertible, which makes it a positive definite matrix (since H is PSD). Since $H_{jj} - H_{I_j j}^T H_{I_j I_j}^{-1} H_{I_j j} > 0$, using Guttman rank additivity formula $H_{J_j J_j} \succ 0$, where $J_j = I_j \cup j$. Since $H_{J_j' J_j'}$ is a submatrix of $H_{J_j J_j}$, it is positive definite and hence its schur complement $H_{jj} - H_{I_j' j}^T H_{I_j' I_j'}^{-1} H_{I_j' j} > 0$. Thus for all $j \in [n]$, the corresponding D_{jj}'s are well-defined in the new graph $\tilde{\mathcal{G}}$.

Note that $\kappa_{\ell d}^{\tilde{\mathcal{G}}} = \max_{i \in [n-1]} 1/(1 - \beta_i^2) < \max_{i \in \bar{K}} 1/(1 - \beta_i^2) = \kappa_{\ell d}^{\mathcal{G}}$, for tridiagonal graph, where $\beta_i = H_{ii+1}$, in the case where $H_{ii} = 1$. This is because the $\arg\max_{i \in [n-1]} 1/(1 - \beta_i^2) \in K$.

A.5 Additional Experiments, ablations, and details

Effect of band size in banded-SONew Increasing band size in banded-SONew captures more correlation between parameters, hence should expectedly lead to better preconditioners. We confirm this through experiments on the Autoencoder benchmark where we take band size = 0 (diag-SONew), 1 (tridiag-SONew), 4, and 10.

Effect of Numerical Stability Algorithm 3 On tridiag-SONew and banded-4-SONew, we observe that using Algorithm 3 improves training loss. We present in Table 4 results where we observed significant performance improvements.

Table 4: **bfloat16 experiments on Autoencoder benchmark with and without Algorithm 3.** We observe gains in training loss when using Algorithm 3

Optimizer	Train CE loss - without Algorithm 3	Train CE loss - with Algorithm 3
tridiag-SONew	53.150	51.936
band-4-SONew	51.950	51.84

Table 5: **Large Scale Benchmarks.** We compare tds vs Adam on the following large scale benchmark. We compare train CE loss, and validation performance - measured as precision for OGBG benchmark and error rate for Resnet50 and Vision Transformer benchmark.

Benchmark	# model parameters	# training points	Train Loss (Adam)	Train Loss(SONew)	Valid. perf. (Adam)	Valid. perf. (SONew)
Resnet50-Imagenet	23M	1.2M	0.0951	**0.0857**	22.61%	**22.55**%
Vision Tansformer-Imagenet		1.2M	0.4589	**0.45506**	**22.85**%	23.30%
OGBG-molpcba		437,929	**0.0145**	0.0157	**0.2835**	0.2820

Table 6: **Optimal hyperparamers for Autoencoder Benchmark**

(a) float32 experiments optimal hyperparamters

Baseline	β_1	β_2	ϵ	lr
SGD	0.99	0.91	8.37e-9	1.17e-2
Nesterov	0.914	0.90	3.88e-10	5.74e-3
Adagrad	0.95	0.90	9.96e-7	1.82e-2
Momentum	0.9	0.99	1e-5	6.89e-3
RMSProp	0.9	0.9	1e-10	4.61e-4
Adam	0.9	0.94	1.65e-6	3.75e-3
Diag-SONew	0.88	0.95	4.63e-6	1.18e-3
Shampoo	0.9	0.95	9.6e-9	3.70e-3
tridiag	0.9	0.96	1.3e-6	8.60e-3
band-4	0.88	0.95	1.5e-3	5.53e-3

(b) bfloat16 experiments optimal hyperparamters

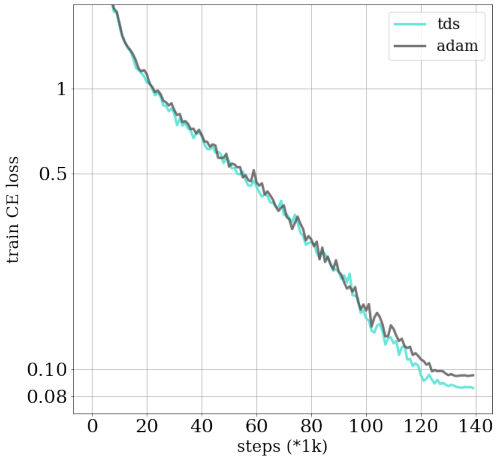
Baseline	β_1	β_2	ϵ	lr
SGD	0.96	0.98	2.80e-2	1.35e-2
Nesterov	0.914	0.945	8.48e-9	6.19e-3
Adagrad	0.95	0.93	2.44e-5	2.53e-2
Momentum	0.9	0.99	0.1	7.77e-3
RMSProp	0.9	0.9	2.53e-10	4.83e-4
Adam	0.9	0.94	3.03e-10	3.45e-3
Diag-SONew	0.9	0.95	4.07e-6	8.50e-3
Shampoo	0.85	0.806	6.58e-4	5.03e-3
ztridiag	0.83	0.954	1.78e-6	7.83e-3
band-4	0.9	0.96	1.52e-6	4.53e-3

Large Scale Benchmark Comparison To test efficacy of SONew in Deep Learning, we test our method against Adam on 3 large scale benchmarks, given in Table 5. We see that SONew performs comparable or outperforms Adam in all the benchmark in both train loss and validation performance. Moreover, we plot the training and validation curve in Figure 2. We observe that SONew has an early advantage over Adam.

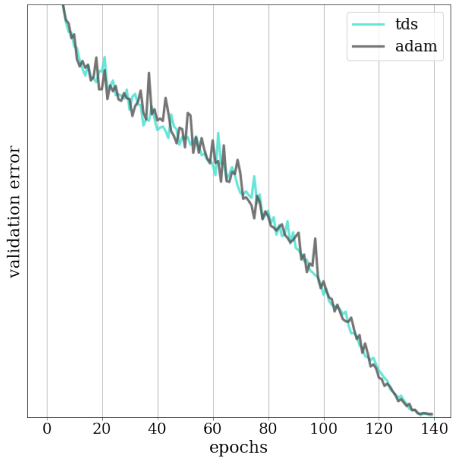
Hyperparaeter search space We provide the hyperparamter search space for experiments presented in Section 6. We search over $2k$ hyperparameters for each experiment using a Bayesian Optimization package. The search ranges are: first order momentum term $\beta_1 \in [1e-1, 0.999]$, second order momentum term $\beta_2 \in [1e-1, 0.999]$, learning rate $\in [1e-7, 1e-1]$, $\epsilon \in [1e-10, 1e-1]$. We give the optimal hyperparameter value for each experiment in Table 6

For large scale benchmark (Table 5), we search $\beta_1, \beta_2 \in [0.1, 0.999]$, $lr \in [1e-5, 1e-1]$, $\epsilon \in [1e-8, 1e-4]$, weight decay $\in [1e-5, 1.0]$, learning rate warmup $\in [2\%, 5\%, 10\%]*$total_train_steps. We use cosine learning rate schedule. For resnet50 imagenet, we also search label smoothing over $[0.0, 0.2]$. Batch size was kept = 1024, 1024, and 512 for Resnet50, Vision Transformer, and OGBG respectively. We sweep over 100 hyperparameters in the search space for both SONew and Adam.

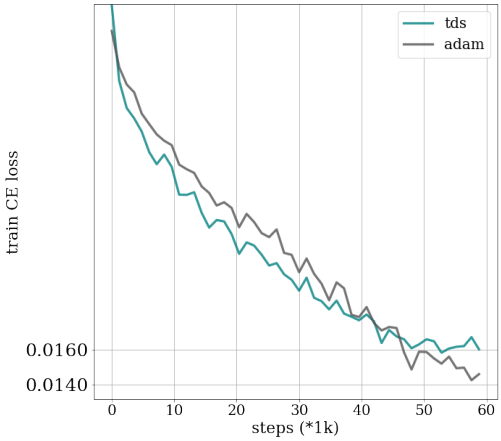
(a) Imagenet-Resnet50 train CE loss



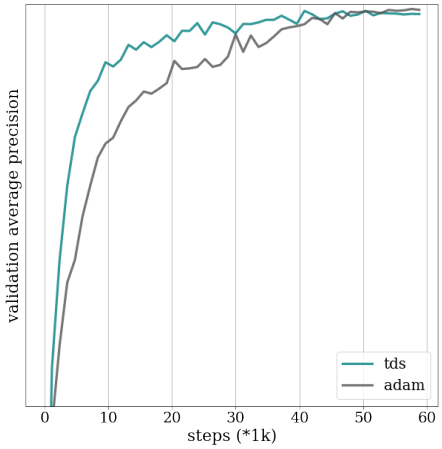
(b) Imagenet-Resnet50 valid. error rate



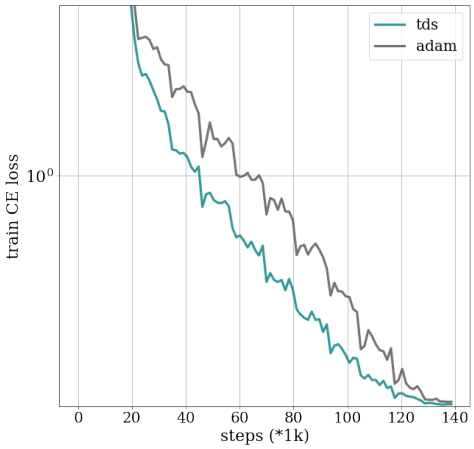
(c) OGBG train CE loss



(d) OGBG valid. average precision



(e) Imagenet-Vision Transformer train CE loss



(f) Imagenet-Vision Transformer validation error rate

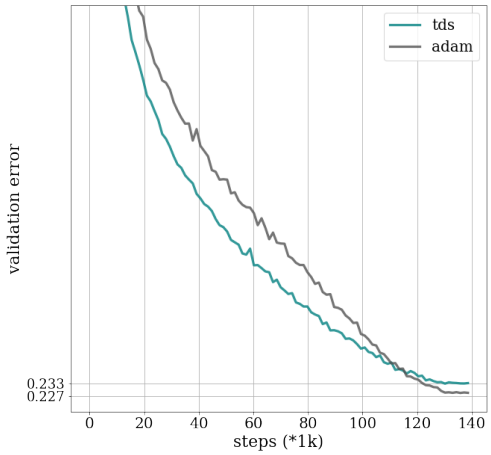


Figure 2: SONew vs Adam on large scale benchmark - Train Cross Entropy (CE) Loss (left column) and validation performance (right column). We see that SONew has an early advantage over Adam in all the experiments.