Scalable Approximate NonSymmetric Autoencoder for Collaborative Filtering

Martin Spišák*
GLAMI
Prague, Czech Republic
martin.spisak@glami.cz

Radek Bartyzal
GLAMI
Prague, Czech Republic
radek.bartyzal@glami.cz

Antonín Hoskovec†
GLAMI
Prague, Czech Republic
antonin.hoskovec@glami.cz

Ladislav Peška
Faculty of Mathematics and Physics,
Charles University
Prague, Czech Republic
ladislav.peska@matfyz.cuni.cz

Miroslav Tůma
Faculty of Mathematics and Physics,
Charles University
Prague, Czech Republic
miroslav.tuma@mff.cuni.cz

ABSTRACT

In the field of recommender systems, shallow autoencoders have recently gained significant attention. One of the most highly acclaimed shallow autoencoders is EASE^R, favored for its competitive recommendation accuracy and simultaneous simplicity. However, the poor scalability of EASE^R (both in time and especially in memory) severely restricts its use in production environments with vast item sets. In this paper, we propose a hyperefficient factorization technique for sparse approximate inversion of the data-Gram matrix used in EASE^R. The resulting autoencoder, SANSA, is an end-to-end sparse solution with prescribable density and almost arbitrarily low memory requirements — even for training. As such, SANSA allows us to effortlessly scale the concept of EASE^R to millions of items and beyond.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Sparse autoencoders, Numerical approximation, Sparse approximate inverse, Algorithm scalability

ACM Reference Format:
Martin Spišák, Radek Bartyzal, Antonín Hoskovec, Ladislav Peška, and Miroslav Tůma. 2023. Scalable Approximate NonSymmetric Autoencoder for Collaborative Filtering. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3604915.3608827

*Also with Faculty of Mathematics and Physics, Charles University.
†Also with Department of Physics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague.

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09.
https://doi.org/10.1145/3604915.3608827

1 INTRODUCTION

Although data sparsity is typically viewed as an obstacle for recommender systems (RS), it presents an opportunity for developing efficient algorithms, e.g., via collaborative filtering (CF). Conceptually, CF methods extract the wisdom of the crowd from a (potentially very sparse) graph representation of the interactions inside the crowd. Over the years, many CF methods have been developed, with the current state-of-the-art dominated by graph neural networks [21, 31, 37, 47] and autoencoders [28, 33, 38, 41]. Among especially popular *shallow* autoencoders [33, 41], the linear model EASE^R [41] is one of the most prospective thanks to its state-of-the-art recommendation accuracy — even compared to deeper models [15], straightforward implementation derived from the closed-form solution, and faster training than deeper models, e.g., [28, 33]. Given training data — a user–item interaction matrix[1] $X \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ — the training of EASE^R proceeds as follows:

(1) Compute a regularized Grammian $A = X^T X + \lambda I$. (This step may be done in the pre-training phase.)
(2) Compute $P = A^{-1}$.
(3) Post-process P (by column scaling) to obtain weights B.

During the inference, the model predicts ratings as $\vec{r}^T = \vec{u}^T B$, where $\vec{u}$ is the input vector of the user's feedback.

Unfortunately, broader adoption of EASE^R in real-world RS is difficult due to the model's poor scalability with respect to the volume of items $|\mathcal{I}|$. Inverting a large matrix (Step 2) can be prohibitively slow for use in production software. More importantly, the learned weight matrix may require too much memory — even if the data-Gram item–item matrix A is very sparse. It is well known that the inverse of an irreducible matrix is fully dense (regardless of the sparsity of the original matrix); see [16, 36]. Since the eventual irreducibility of A stems from the connectedness of the underlying item–item network, we may expect the weight density to cause a problem. As an example, for a dataset with 1 million items, EASE^R would have up to 1 *trillion* parameters, requiring up to 4 TB of memory (using `float32`) during inference; this is far too expensive for many relevant use-cases.

Our paper offers a solution: an accurate sparse approximate model. To facilitate efficient training, we propose a sparse approximate inversion framework with two specific variants depending on

[1]The matrix is typically large, sparse, and overdetermined ($|\mathcal{U}| \gg |\mathcal{I}|$).

training-time resource availability. When training time, rather than memory, is limiting, we propose a fast (but more memory-intensive) training procedure by exploiting the sparsity or supernodal structure of A. Otherwise, if memory is the bottleneck (as common in large-scale production RS), we utilize the sparsity of item associations for a fast, end-to-end sparse training with almost arbitrarily low memory overhead. This allows us to effortlessly scale the concept of EASE^R to millions of items and beyond.

1.1 Related work

On domains with vast item sets, direct data about the relation between a random item pair is statistically unlikely to exist (X^TX is sparse; see Section 4 for an example). Since both users and items may have only a few interactions, locally-focused CF methods [21, 31, 47] may struggle with long-tail items and niche users. The power of EASE^R stems from its ability to find important *long-distance* item relations via long chains of users [41], making it very desirable for use in large-scale settings. However, its difficult scalability poses a severe obstacle, motivating several modifications [24, 40, 42–44, 46]. For conciseness, we only discuss the two variants focusing on efficient, scalable training.

ELSA [46] approximates the weight matrix of EASE^R as the Grammian of a dense low-rank weight matrix optimized by gradient descent. It improves both training time and memory complexity over basic EASE^R and even achieves slightly higher accuracy on smaller datasets — likely due to the regularization effect of rank reduction. However, low-rank approximation limits the expressiveness of the autoencoder; see, e.g., [9]. This can negatively affect recommendation quality [40, 42], especially on domains with extensive long-tail, where dependencies may be crucial despite being subtle.

MRF [42] method learns a sparse *full-rank* approximation of A^{-1} from a Markov Random Field (hence the name MRF). The method estimates A^{-1} through inverses of leading clusters interconnected by interfaces (see, e.g., [2, 17] for more details), i.e., a specific domain decomposition where the interfaces represent overlaps [26, 39]. Crucially, MRF closely matches the performance of dense EASE^R even at high compression rates and allows for a trade-off between model size, training time, and recommendation accuracy. The approach demonstrates that exploiting data sparsity is the key to training large-scale CF models efficiently and proves the viability of scaling EASE^R-like methods to domains with vast item sets, where X^TX is sparse. Unfortunately, the training of MRF may still require significant resources. The sparsity pattern in MRF is estimated from a *dense* item correlation matrix, which can be very large[2]. Even inverting small matrices may be memory costly if their number is too large — they must be kept in memory simultaneously (which is expensive) or progressively loaded (slow). In contrast, our proposal can extract relations outside leading clusters and, more importantly, train end-to-end sparsely, resulting in substantial efficiency gains.

2 MODEL ARCHITECTURE

As in [42], we propose a *full-rank* sparse modification of EASE^R. The final model weights arise from the sparse approximation of $A^{-1} = (X^TX + \lambda I)^{-1}$ by applying the original post-processing (scaling). Motivated by memory-critical problems, we design the architecture so that the maximum density of weights can be *selected* by a parameter, and training memory can be restricted. The model converges to EASE^R as the allowed density increases.

Contemporary matrix inversion research [5, 11, 19, 36, 45] implies that sparse approximate inverses extract dominant information reliably and that finding the (right) inverse of A is equivalent to solving $|\mathcal{I}|$ independent problems $A\vec{m}_j = \vec{e}_j$ with the complexity dominated by the LU decomposition of A. Sparse LU factorization is well-understood, but it may suffer from instabilities that are hard to cure in parallel environments and the constraining sequentiality of the (fan-out) back substitution. However, the LU factorization for a symmetric positive definite (SPD) matrix A reduces to Cholesky factorization, which a) is unconditionally backward stable, b) with well-parallelizable fan-in phase, and c) avoids the fan-out phase chokepoint. Our idea is then to reduce the problem of sparse approximation of A^{-1} into two simpler problems:

(1) Find a sparse approximate square-root-free Cholesky decomposition $A \approx \widehat{L}\widehat{D}\widehat{L}^T$.
(2) Find a sparse approximate inverse K of the matrix $\widehat{L}$.

The resulting factorization $K^T\widehat{D}^{-1}K$ is used to build the encoder layer W^T and the decoder layer Z of our Scalable Approximate Non-Symmetric Autoencoder (SANSA). Formally, SANSA approximates the encoder–decoder matrix B of EASE^R using a two-layer linear model with no activation:

$$B \approx W^TZ.$$

We illustrate the architecture of SANSA in Fig. 1. The inference is analogous to the original. For a user's feedback vector $\vec{u}$, the ratings are obtained by two sparse matrix–vector multiplications, $\vec{r}^T = (\vec{u}^TW^T)Z$. Note that W^TZ approximates B *before* forcing its diagonal to zero. This constraint is merely a residual connection[3] which may be imposed by masking out input items in the prediction if self-similarity is prohibited.

Implicitly representing A^{-1} as $K^T\widehat{D}^{-1}K$ yields a much denser operator with equal storage cost [5]. Such compression is advantageous in RS: compared to sparse single-layer models (such as MRF), a sparse linear model with two layers and the same number of parameters will generate denser prediction vectors from sparse input interactions, i.e., recommend more items. Additionally, factorized approaches are able to circumvent the part of training memory overhead stemming from the computation of $X^TX+\lambda I$, as Cholesky factorization can use this matrix implicitly from X [7].

3 MODEL TRAINING

Our attention now turns to specific methods used to train SANSA. We then describe the training procedure and explain how our factorized approach implies a reasonable training loss.

[2]The matrix has $|\mathcal{I}|^2$ entries. E.g., for Amazon Books [32] with $|\mathcal{I}| = 91599$, the matrix requires 33.6 GB (in `float32`) — and item sets can be *much* larger.

[3]Let B be the weight matrix of EASE^R before applying the diagonal constraint and let $\vec{b} = \mathrm{diag}(B)$ and $C = B - \mathrm{diag}(\vec{b})$ (that is, C is obtained from B by setting its diagonal entries to zero). The prediction computes $\vec{p}^T = \vec{u}^TC = \vec{u}^TB - \vec{u}^T\mathrm{diag}(\vec{b})$. But $\vec{b} = -\vec{1}$, since B is computed by scaling the columns of A^{-1} by the reciprocals of their (negative) diagonal entries. That is, $\vec{p}^T = \vec{u}^TC = \vec{u}^TB - \vec{u}^T(-I) = \vec{u}^T(B+I)$.

3.1 On computation of sparse approximate inverses

An essential advantage of the mentioned approximate inverse research [1, 5, 8, 36] (compared to, e.g., MRF) is the ability to consider *non-local* dependencies in addition to the closely connected clusters, and detect such links automatically. While more possible ways exist [5], we focus on highly parallel, scalable approaches developed for high-performance computing (HPC). As discussed earlier, we target factorizations of the form $A^{-1} \approx \widehat{M} = K^T \widehat{D}^{-1} K$, which are reliable since A is SPD. Let us emphasize the following. We are looking for $\widehat{M}$ which *operates* (when applied from the right) as closely to A^{-1} as possible, i.e., formally, we want $\widehat{M}$ to minimize $\|I - AM\|_F{}^4$ — this is a different problem than minimizing $\|\widehat{M} - A^{-1}\|$; see [11]. Algorithmically, for $A = LDL^T$, we first compute its sparse approximate decomposition $\widehat{L}\widehat{D}\widehat{L}^T$ with small $\|L - \widehat{L}\|_F$ and $\|D - \widehat{D}\|_F$, and then find sparse $K \approx \widehat{L}^{-1}$ such that $\|I - \widehat{L}K\|_F$ is small. The final product $K^T \widehat{D}^{-1} K$ is a good sparse approximation of A^{-1} since $\|I - A(K^T \widehat{D}^{-1} K)\|_F = \|I - (LDL^T)(K^T \widehat{D}^{-1} K)\|_F$ is (relatively) small. While specific approaches for obtaining K and $\widehat{D}$ may not be optimal, the computed approximation converges to A^{-1} as the allowed density increases.

3.1.1 Methods for approximate Cholesky factorization. Modern sparse Cholesky factorization starts by finding a permutation that (approximately) minimizes fill-in and enhances parallelizability.[5] Besides lowering memory requirements, this also reduces information loss in the a posteriori sparsification. Approximation may be applied before, during, and after the factorization. Sparsification before factorization is often suboptimal since it may lose subtle relations among data, as clearly shown, e.g., in a structural mechanics problem [4]. However, it may help decrease the time and memory requirements if a large matrix strongly fills. A better way is to sparsify the (possibly filled in) exact factor L. The use of Frobenius norm guarantees that zeroing entries smallest in magnitude in L delivers the best approximate factor $\widehat{L}$ with a given density. Moreover, factors $\widehat{L}$ converge monotonically to L with increasing density (i.e., denser $\widehat{L}$ are better).

Unfortunately, the amount of generated fill-in (that needs to be stored) may limit the applicability of this variant for denser input matrices X. Luckily, the symbolic factorization phase computes the size of the (complete) factor L in nearly linear complexity [36] and can provide a warning. Then, if memory is limiting, it is possible to sparsify *during* factorization using incomplete Cholesky factorization (ICF) — e.g., [29] based on Crout form of the algorithm, with practically linear complexity for close to uniform distribution of nonzeros in A [25]. It operates with *prescribed, almost arbitrarily small* memory overhead, but with some trade-offs: 1) incomplete factorization may break down; additional regularization of A may be needed to prevent this, and 2) compared to the a posteriori sparsified complete factorization, the convergence to the exact factor L may not be monotone.

Summarizing, users of SANSA can select from two sparse (approximate) Cholesky factorization variants. When training memory is not severely limiting, the preferable method is SANSA (CHOLMOD), which uses the (numerically) exact block column code CHOLMOD [10] and a posteriori sparsification to the target density. When training memory is restrictive (as in domains with large item sets), we can use SANSA (ICF), which constructs $\widehat{L}$ by sparsification on the fly using ICF [29]. For SANSA (ICF), users may select to compute denser $\widehat{L}$ to help stabilize the factorization (the factor is sparsified to the target density afterward). Unsurprisingly, when ICF is used for a denser A, we should expect a less accurate approximation due to the loss of information throughout the incomplete factorization.

3.1.2 Approximate inverse of $\widehat{L}$ and training loss. As mentioned above, there are more ways to compute K. We select a parallelizable approach based on minimizing the Frobenius norm of the residual matrix $R = I - \widehat{L}K$ (the minimization agrees with the objective function; see Section 3.1). Specifically, we use a modified Minimal Residual (MR) algorithm [12]. Conveniently, MR finds the dominant sparsity pattern automatically[6]. We use global sparsification instead of the standard column-by-column one to allow for nonhomogenous patterns. The minimization first adjusts the entire matrix using progressively finer *scans*, after which it *finetunes* the columns with highest residual norms — hence the *Uniform* Minimal Residual (UMR) algorithm. Each iteration first computes R and selects columns for modification. The matrix R also shows an explainable training loss: the mean of squared column norms of R is equal to $n \cdot \mathrm{MSE}(I, \widehat{L}K)$, or equivalently $\frac{\|I - \widehat{L}K\|_F^2}{\|I\|_F^2}$ — the relative residual norm squared. Individual iterations optimize "batches" of columns to manage memory requirements.

Since $\widehat{L}$ is unit lower triangular, we propose a better choice of the initial guess for (U)MR. Inverting just the signs of the subdiagonal entries of $\widehat{L}$ gives $K^{(0)} = 2I - \widehat{L}$. This guess is essentially for free and corresponds to a step of Schultz iterative process [35] with initial guess $K^{(0)}_{Sch.} = I$. One step of this process from the initial guess I *exactly inverts* matrices of form $I + N$, where N is a strictly (lower, or upper) triangular matrix such that $N^2 = 0$. When $\widehat{L}$ is sparse, few subdiagonal entries of $\widehat{L}$ exist and are mostly small in magnitude. Hence, the subdiagonal part of $\widehat{L}$ (denoted N) satisfies $N^2 \approx 0$. The initial guess is also closely linked to column elimination matrices and their inverses [18]. Therefore, the initial guess $2I - \widehat{L}$ can be very close to $\widehat{L}^{-1}$, needing only minor refinement, as verified in our experiments (Section 4).

3.2 Training procedure

The only compute-intensive part is the encoder training performed using the factorized sparse inversion. The training begins by finding a suitable item permutation[7] represented by matrix P. Then a sparse approximate LDL^T factorization of PAP^T is computed. We get a nonsingular diagonal matrix $\widehat{D}$ and a unit lower triangular matrix $\widehat{L}$ such that $\widehat{L}\widehat{D}\widehat{L}^T \approx PAP^T$.

To obtain the approximate inversion, we invert $\widehat{D}$ and compute an approximate inverse of $\widehat{L}$ in two steps. First, we take the "free" initial guess $K^{(0)} = 2I - \widehat{L} \approx \widehat{L}^{-1}$. If $K^{(0)}$ is not accurate enough, we refine it using the UMR algorithm, which performs a given number

[4] The choice of Frobenius norm matches the objective function of EASE^R.
[5] The complexity of this search is loosely bounded by $O(\texttt{density}(A) \times |\mathcal{I}|^3)$ [22].
[6] Recall that MRF relies on the computation of a dense statistical matrix to find its sparsity pattern, see Section 1.1.
[7] Our codes use COLAMD [14].

input user–item interaction matrix X, L2 regularization λ
compute sparse $LDL^T \approx P(X^TX + \lambda I)P^T$ (for a permutation P)
compute sparse $K \approx L^{-1}$
$W \leftarrow KP$
$Z_0 \leftarrow D^{-1}W$
$\vec{r} \leftarrow \mathrm{diag}(W^T Z_0)$
$Z \leftarrow$ scale the columns of Z_0 by $-1/\vec{r}$
return W^T, Z

Algorithm 1: The training procedure of SANSA is based on factorized sparse approximate inversion. The final scaling is applied to the decoder only.

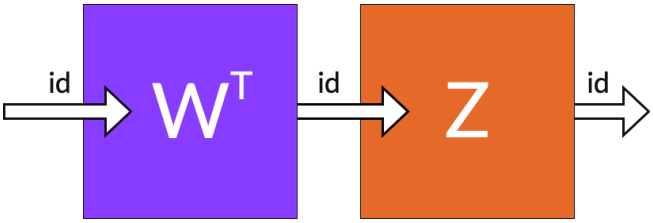

Figure 1: SANSA is a sparse nonsymmetric encoder–decoder model. To disallow recommending input items, we mask the prediction vector, or add an input–output residual connection.

of scans and finetune steps[8]. Once the iteration process ends, we have a sparse $K \approx \widehat{L}^{-1}$, a dense vector representing $\widehat{D}^{-1}$, and a vector representing P. The encoder layer of SANSA, the matrix W^T, is obtained by transposing the column-permuted approximate inverse K. Note that W^T is no longer lower triangular. To summarize, the approximation of A^{-1} is expressed using the encoder W^T and the diagonal matrix $\widehat{D}^{-1}$ as

$$W^T\widehat{D}^{-1}W = P^TK^T\widehat{D}^{-1}KP \approx P^T\widehat{L}^{-T}\widehat{D}^{-1}\widehat{L}^{-1}P \approx A^{-1}.$$

Finally, we apply column scaling — corresponding to scaling the j-th column of A^{-1} by $-A_{jj}^{-1}$ in the original model — to obtain the approximation of the weights B. Scaling only the decoder layer is sufficient (hence, SANSA is a *nonsymmetric* autoencoder): $Z = -(D^{-1}W)\mathrm{diag}(W^T\widehat{D}^{-1}W)^{-1}$. We summarize the training procedure in Algorithm 1.

4 EXPERIMENTS

In the previous sections, we proposed methods for constructing sparse approximations of EASE^R for inference with limited memory. The two methods suit different RS scenarios, depending on whether *training* memory limitations play a decisive role. We conduct experiments on popular music and book datasets to address the two scenarios. For logical coherence, we validate the robustness of our approach in Section 4.1.1 before demonstrating our main contribution — the unparalleled scalability of SANSA (ICF) to large domains — in Section 4.1.2.

[8]All UMR iterations first compute $R = I - \widehat{L}K^{(i)}$, where $K^{(i)}$ is the current approximation of $\widehat{L}^{-1}$. R is also used to calculate the training loss.

First, we evaluate the accuracy and efficiency of the a posteriori sparsified (CHOLMOD) variant. We selected the Million Song Dataset (**MSD**) [6] for this test since it is with 41 140 items among the largest benchmark datasets for CF. However, it is still small enough to allow the complete sparse factorization on a moderately sized `m6i.4xlarge` instance with 64 GB RAM. Most importantly, MSD challenges the robustness of sparse approaches because it is densely connected: although the interaction density is 0.14%, the item–item matrix A is 41.54% dense. We use the established preprocessing and evaluation protocol of [28] for reproducibility and evaluate accuracy using the same metrics, i.e., Recall@20 (r@20), Recall@50 (r@50), and nDCG@100 (n@100). For baselines, we trained dense EASE^R [41] using the same L2 regularization as SANSA (CHOLMOD), as well as the other sparse approach - MRF [42]. For context, we also include the results of additional relevant baselines reprinted from [28] and [38]. These include: deep variational autoencoder RECVAE [38], ranked second on MSD according to their paper; linear *low-rank* factorization model WMF [23]; multinomial variational autoencoder MULT-VAE^{PR} [28].

To demonstrate the scalability of SANSA (ICF), we selected **Amazon Books** [32], the largest and sparsest commonly used benchmark with 91 599 items, interaction density 0.062%, and item–item matrix density 3.94%, hence nearest to the intended real-world use of SANSA. We follow the preprocessing and evaluation protocol of the well-known benchmark [13] and evaluate not only recommendation accuracy (namely, Recall@20 (r@20) [9] and nDCG@20 (n@20)) and model compression (with respect to dense EASE^R), but also training time (using `time.perf_counter`) and memory requirements (using the `memory-profiler` module). As for baselines, we compare SANSA (ICF) with MRF and the results of other state-of-the-art models reprinted from [13], namely: EASE^R [41] (which we did not reproduce ourselves due to high training cost); linear model SLIM [33] — current state-of-the-art on Amazon Books according to [13]; item-oriented neighborhood-based method ITEMCF [34], ranked second in [13]; graph convolutional network ULTRAGCN [31], ranked second among graph convolution methods in [13].

All codes used in experiments (including configs), results, and logs are available at https://github.com/glami/sansa/.

4.1 Results

4.1.1 Robustness and efficiency. The results in Table 1 demonstrate the robustness of SANSA: if we allow sufficient weight density and train the model long enough, SANSA can achieve the same accuracy as dense EASE^R. Table 1 and Fig. 2 also illustrate the monotonic convergence for the a posteriori sparsified (CHOLMOD) variant — the approximation quality improves with increased density. Even very sparse models can approximate EASE^R with high accuracy, despite the high density of item–item links in MSD; 50 to 1 000 times sparser full-rank models perform on par with EASE^R. [10]

[9]Note that [13] and [28] use different definitions of recall. In each experiment, we use the recalls used in corresponding evaluation protocols.

[10]We observed similar behavior on other dense CF datasets — Goodbooks-10k [48], MovieLens-20M [20], and Netflix Prize [3]. Interestingly, both MRF and SANSA (CHOLMOD) slightly outperform state-of-the-art EASE^R on Goodbooks-10k, hinting at a possible beneficial effect of sparse modeling on recommendation accuracy in dense domains, as argued by [40].

		MSD			
d%	model	r@20	r@50	n@100	time
0.1	MRF ($r = 0$)	0.330	0.421	0.385	64 s
	MRF ($r = 0.5$)	0.326	0.417	0.380	55 s
	SANSA (CH.)	0.328	0.422	0.383	200 s
	SANSA (ICF)	0.288	0.385	0.346	190 s
0.5	MRF ($r = 0$)	0.333	0.427	0.389	183 s
	MRF ($r = 0.5$)	0.329	0.424	0.384	90 s
	SANSA (CH.)	0.331	0.426	0.387	253 s
	SANSA (ICF)	0.276	0.370	0.337	632 s
2.0	MRF ($r = 0$)	0.333	0.428	0.390	1031 s
	MRF ($r = 0.5$)	0.329	0.426	0.385	457 s
	SANSA (CH.)	0.332	0.427	0.388	502 s
	SANSA (ICF)	0.298	0.399	0.359	528 s
	EASE^R	0.332	0.428	0.388	312 s
results reprinted from [38] and [28]					
	RECVAE	0.276	0.374	0.326	——
	WMF	0.257	0.312	0.257	——
	MULT-VAE^{PR}	0.266	0.364	0.316	——

Table 1: Highly compressed SANSA (CHOLMOD) and MRF models achieve accuracy comparable to EASE^R even on dense datasets. As the number of nonzeros in the approximation increases, imposing sparsity via masking (MRF) becomes a performance bottleneck; SANSA uses efficient sparse operations which scale better. The standard error in accuracy measurements is about 0.001.

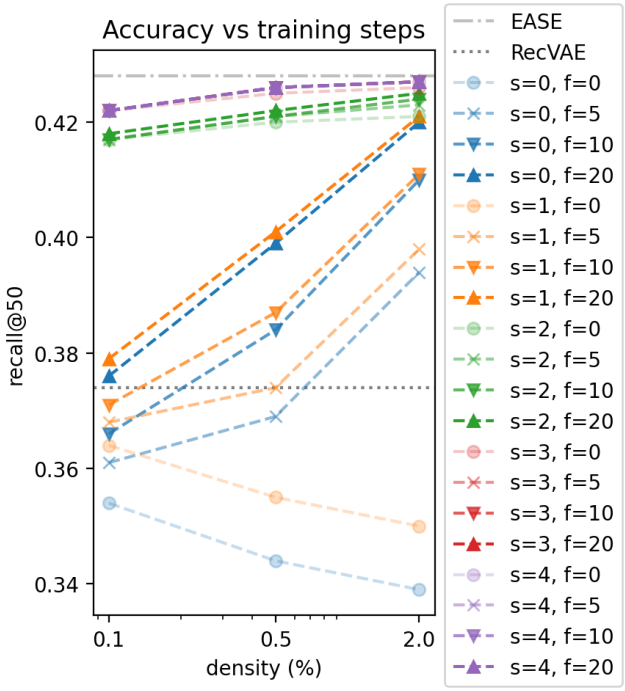

Figure 2: Accuracy of SANSA (CHOLMOD) on MSD after various number of training scans s and finetune steps f.

In larger domains, it may be desirable to trade recommendation accuracy for shorter training. For example, MRF uses parameter r to prune dependencies between item clusters. SANSA provides a similar (although less interpretable) possibility: applying fewer UMR iterations yields a coarser approximation. To analyze the trade-off, we compared checkpoints of SANSA (CHOLMOD) trained for different numbers of UMR scans s and finetune steps f against two baselines: EASE^R to measure the distance from the uncompressed model, and a deep variational autoencoder RECVAE [38], ranked second on MSD according to [38]. The results in Fig. 2 show that short training can, too, produce a close approximation of the dense EASE^R. Notably, we can train very sparse models using but a few short UMR iterations and obtain performance close to state-of-the-art. We discuss how sparsity helps this trade-off in Section 4.1.2. For completeness, SANSA (ICF) performed about 6-17% worse than SANSA (CHOLMOD) on MSD, but this is to be expected: incomplete factorization of a dense matrix loses information during computation and requires more robust regularization to prevent breakdowns. For example, the breakdowns at 0.5% density forced restarts with additional diagonal shifts, further decreasing accuracy and increasing training time. Nevertheless, various early checkpoints of SANSA (CHOLMOD) (Fig. 2) and even the ICF variant outperform other models on MSD. We conclude that very sparse or coarse approximations of EASE^R can be competitive yet very cheap and practical (e.g., for ensemble models).

Compared with MRF, in terms of accuracy, SANSA (CHOLMOD) performs in between MRF ($r = 0$) and pruned MRF ($r = 0.5$) on all tested density levels (d%), see Fig. 2. [11] At $d = 0.1$%, imposing the computed sparsity pattern is inexpensive, and MRF trains three to four times faster than SANSA (CHOLMOD). However, masking operations on sparse matrices do not scale well as the number of nonzeros increases [12]. As a result, training MRF becomes expensive as the total number of nonzero elements in the approximation increases, since r-pruning does not help with this problem. At $d = 2$%, SANSA (CHOLMOD) trains almost as fast as the pruned MRF ($r = 0.5$). Lastly, vectors predicted by a *two-layer* SANSA are significantly denser. Hence, SANSA can recommend more items from sparse inputs than MRF, which may be desirable in practice.

4.1.2 Extreme scalability. Restrictive memory limitations of a very large item–item network can be avoided through its inevitable sparsity (i.e., X^TX is sparse). In such cases, L is likely sparse, too, and $\widehat{L}$ computed by ICF should be close to L. At the same time, for very sparse $\widehat{L}$, the free initial guess $2I - \widehat{L}$ is very close to the exact $\widehat{L}^{-1}$, needing little to no refinement. As a result, our method essentially reduces the sparse approximate inversion to a cheaply obtained incomplete factorization. This shortcut fundamentally reduces training time and memory requirements.

Therefore, it is by no surprise that on Amazon Books, SANSA (ICF) with 0.84 million parameters (10 000 times compressed with respect to dense EASE^R) *trains orders of magnitude faster* than any other non-sparse state-of-the-art method (dense autoencoders, nearest

[11] A more accurate comparison is difficult, as MRF uses additional data normalization to tackle popularity bias, see [42] and Section 1.1.

[12] Compared with that, SANSA performs its training using Cholesky factorization, sparse matrix–matrix multiplications, sparsifications and element-wise operations working on contiguous memory, and (block) column manipulations - all efficient sparse matrix operations.

	Amazon Books						
				results reprinted from [13]:			
	SANSA (ICF)	MRF ($r = 0$)	MRF ($r = 0.5$)	EASE^R	SLIM	ITEM-CF	ULTRA-GCN
r@20	**0.077**	0.071	0.069	0.071	0.075	0.074	0.068
n@20	**0.064**	0.058	0.055	0.057	0.060	0.061	0.056
TRAINING RESOURCES							
vCPU	**2**	16	16	28	28	28	20*
memory usage (GB):							
peak	**9.18**	96.45	96.58	—— not measured; costly ——			
avg.	**3.87**	49.12	49.75	—— not measured; costly ——			
time	**49 s**	172 s	167 s	222 m	316 m	57 m	45 m

*and a GPU (RTX 2080)

Figure 3: Thanks to end-to-end sparsity, training SANSA (ICF) on 2 vCPUs takes about 3 times less than MRF on 16 vCPUs and orders of magnitude less than non-sparse model training. the training of SANSA requires minuscule memory - unparalleled even with MRF. As a bonus, it also achieves new state-of-the-art accuracy. the standard error in accuracy measurements is about 0.0005.

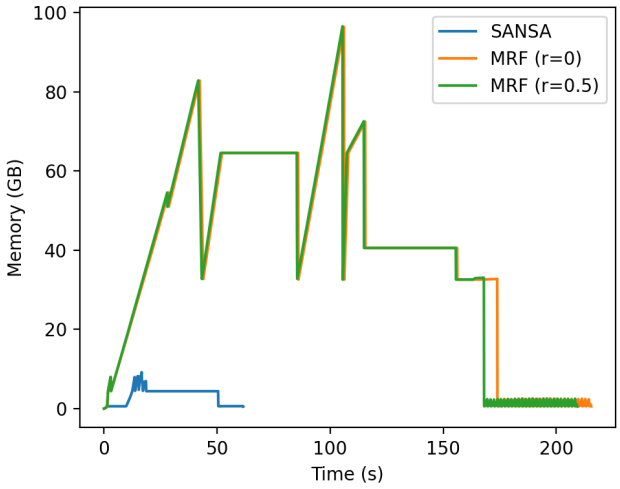

Figure 4: Comparison of time and memory usage of SANSA (ICF) versus MRF on Amazon Books. The final flatline on each graph corresponds to the evaluation.

neighbors approaches or graph neural networks); see Table 3. Furthermore, SANSA (ICF) trains more than three times faster than MRF — partially due to the discussed performance bottleneck caused by masking but also due to the large number of leading clusters requiring inversion. Moreover, this speedup was achieved on a single-core `r6i.large` instance (2 vCPUs, 16 GB RAM), which is much smaller compared to `r6i.4xlarge` with eight cores (16 vCPUs, 128 GB RAM) needed for MRF [13]. As such, training SANSA (ICF) is much more cost-effective compared to other models, e.g., in terms of total FLOPs. In addition, thanks to efficient sparse operations, the training requires merely one tenth of the memory required for MRF training (Table 3, Fig. 4), and this edge can be improved further: our code keeps up to two copies of X^TX in memory, amounting to about 8 GB for Amazon Books. This overhead can be eliminated by implicitly constructing X^TX during ICF (see Section 2). For perspective, SANSA (ICF) for Amazon Books could then be trained on a Raspberry Pi or a smartphone.

While beating EASE^R in terms of accuracy was never our goal, we surpassed its reported performance on Amazon Books by a non-trivial margin and, with it, the current state-of-the-art according to [13]. Since MRF does not outperform EASE^R in our experiment, we do not attribute the accuracy improvement to the regularization effect of sparse approximation discussed by [40]. However, SANSA (ICF) differs from EASE^R (and SANSA (CHOLMOD) and MRF) in scaling (and, hence, also in the used L2 regularization). The scaling is necessary to stabilize the incomplete factorization (see [29] and Section 4.1.1); we leave the analysis of its effect on recommendation and its interpretation for future research.

5 CONCLUSION AND FUTURE WORK

Compared with other approaches, EASE^R takes advantage of long-distance information, crucial for accurate and diverse CF modeling, but expensive to extract and store on vast domains. We propose a solution to this deal-breaking bottleneck using contemporary numerical methods for sparse approximate inversion. Modern approximate inversion allows for the extraction of long-distance information to improve over principally local methods like MRF or graph convolutions. Moreover, sparse approximate inverses can reliably find dominant model information and offer strong compression, further improved by factorization. Where previous approaches lack efficiency in attempting to overpower the problem using operations tailored to dense structures, our end-to-end sparse method is tightly coupled with characteristics of the considered task. Hence, the resulting model, SANSA, trains faster and with minuscule memory requirements. Finally, the method implies a coherent training loss and a simple set of hyperparameters for straightforward production use. Thanks to these properties, SANSA provides a robust yet attainable baseline for researchers with limited resources and large-scale industry environments.

As the next step, we plan to test SANSA in large-scale online experiments. A more long-term goal is further improving the method. In particular, we plan to add tree parallelism based on a nested dissection approach for reordering [27, 30], which should provide an additional boost to scalability. Tree parallelism can mitigate the sequential nature of ICF and result in an HPC-class training algorithm suitable even for the most extensive CF tasks. Interestingly, the nested dissection should also reveal an "arterial structure" of the item–item network. Understanding this structure and the latent embeddings produced by SANSA will benefit interpretability and enable new uses for, e.g., knowledge distillation.

ACKNOWLEDGMENTS

This paper has been supported by Charles University grant SVV-260698/2023. Source codes and raw results can be obtained from https://github.com/glami/sansa/.

[13] MRF cannot be tested on a smaller instance due to training memory requirements approaching 100 GB. Instances used in [13] are even larger, with up to 28 vCPUs and 500+ GB RAM, or 20 vCPUs and a GPU.

REFERENCES

[1] Hartwig Anzt, Thomas K. Huckle, Jürgen Bräckle, and Jack Dongarra. 2018. Incomplete sparse approximate inverses for parallel preconditioning. *Parallel Computing. Systems & Applications* 71 (2018), 1–22. https://doi.org/10.1016/j.parco.2017.10.003

[2] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. 2008. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *J. Mach. Learn. Res.* 9 (2008), 485–516. http://dblp.uni-trier.de/db/journals/jmlr/jmlr9.html#BanerjeeGd08

[3] James Bennett and Stanley Lanning. 2006. The Netflix Prize. *Proceedings of KDD Cup and Workshop* Vol. 2007 (11 2006).

[4] M. Benzi, R. Kouhia, and M. Tuma. 2001. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Comput. Methods Appl. Mech. Engrg.* 190, 49-50 (2001), 6533–6554.

[5] M. Benzi and M. Tůma. 1999. A comparative study of sparse approximate inverse preconditioners. *ANM* 30, 2-3 (1999), 305–340.

[6] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, Anssi Klapuri and Colby Leider (Eds.). University of Miami, Miami, FL, USA, 591–596. http://ismir2011.ismir.net/papers/OS6-1.pdf

[7] Å. Björck. 1996. *Numerical methods for Least Squares Problems.* SIAM, Philadelphia.

[8] Matthias Bollhöfer and Yousef Saad. 2006. Multilevel preconditioners constructed from inverse-based ILUs. *SIAM Journal on Scientific Computing* 27, 5 (2006), 1627–1650. https://doi.org/10.1137/040608374

[9] Ting Chen, Ji Lin, Tian Lin, Song Han, Chong Wang, and Denny Zhou. 2018. Adaptive Mixture of Low-Rank Factorizations for Compact Neural Modeling. https://openreview.net/forum?id=r1xFE3Rqt7

[10] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* 35, 3, Article 22 (oct 2008), 14 pages. https://doi.org/10.1145/1391989.1391995

[11] E. Chow and Y. Saad. 1997. Experimental study of *ILU* preconditioners for indefinite matrices. *J. Comput. Appl. Math.* 86, 2 (1997), 387–414.

[12] Edmond Chow and Yousef Saad. 1998. Approximate Inverse Preconditioners via Sparse-Sparse Iterations. *SIAM J. Sci. Comput.* 19, 3 (may 1998), 995–1023. https://doi.org/10.1137/S1064827594270415

[13] The BARS Community. 2023. BarsMatch: A Benchmark for Candidate Item Matching. https://openbenchmark.github.io/BARS/candidate_matching/

[14] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. 2004. A Column Approximate Minimum Degree Ordering Algorithm. *ACM Trans. Math. Softw.* 30, 3 (sep 2004), 353–376. https://doi.org/10.1145/1024074.1024079

[15] Yushun Dong, Jundong Li, and Tobias Schnabel. 2023. When Newer is Not Better: Does Deep Learning Really Benefit Recommendation From Implicit Feedback?. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 942–952. https://doi.org/10.1145/3539618.3591785

[16] I. S. Duff, A. M. Erisman, C. W. Gear, and J. K. Reid. 1988. Sparsity structure and Gaussian elimination. *ACM SIGNUM Newslett.* 23, 2 (1988), 2–8.

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2007. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9, 3 (12 2007), 432–441. https://doi.org/10.1093/biostatistics/kxm045 arXiv:https://academic.oup.com/biostatistics/article-pdf/9/3/432/17742149/kxm045.pdf

[18] G.H. Golub and C.F. Van Loan. 2013. *Matrix Computations* (fourth ed.). Johns Hopkins University Press, Baltimore and London.

[19] M. J. Grote and T. Huckle. 1997. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing* 18, 3 (1997), 838–853.

[20] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (dec 2015), 19 pages. https://doi.org/10.1145/2827872

[21] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. arXiv:2002.02126 [cs.IR]

[22] P. Heggernes, S. Eisenstat, G. Kumfert, and A. Pothen. 2001. *The computational complexity of the minimum degree algorithm.* ICASE Report No. 2001-42. Insitute for Computer Applications in Science and Engineering, Hampton, Virginia.

[23] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM '08)*. IEEE Computer Society, USA, 263–272. https://doi.org/10.1109/ICDM.2008.22

[24] Olivier Jeunen, Jan Van Balen, and Bart Goethals. 2022. Embarrassingly Shallow Auto-Encoders for Dynamic Collaborative Filtering. *User Modeling and User-Adapted Interaction* 32, 4 (sep 2022), 509–541. https://doi.org/10.1007/s11257-021-09314-7

[25] M. T. Jones and P. E. Plassmann. 1995. An Improved Incomplete Cholesky Factorization. *ACM Trans. Math. Software* 21, 1 (1995), 5–17.

[26] Guy Antoine Atenekeng Kahou, Laura Grigori, and Masha Sosonkina. 2008. A partitioning algorithm for block-diagonal matrices with overlap. *Parallel Comput.* 34, 6-8 (2008), 332–344. https://doi.org/10.1016/j.parco.2008.01.004

[27] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (1998), 359–392. https://doi.org/10.1137/S1064827595287997 arXiv:https://doi.org/10.1137/S1064827595287997

[28] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. https://doi.org/10.1145/3178876.3186150

[29] Chih-Jen Lin and Jorge J. Moré. 1999. Incomplete Cholesky Factorizations with Limited Memory. *SIAM J. Sci. Comput.* 21 (1999), 24–45.

[30] Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. 1979. Generalized Nested Dissection. *SIAM J. Numer. Anal.* 16, 2 (1979), 346–358. https://doi.org/10.1137/0716027 arXiv:https://doi.org/10.1137/0716027

[31] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 1253–1262. https://doi.org/10.1145/3459637.3482291

[32] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 188–197. https://doi.org/10.18653/v1/D19-1018

[33] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining (ICDM '11)*. IEEE Computer Society, USA, 497–506. https://doi.org/10.1109/ICDM.2011.134

[34] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. Association for Computing Machinery, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[35] G. Schulz. 1933. Iterative Berechnung der reziproken Matrix. *ZAMM* 13 (1933), 57–59.

[36] J. A. Scott and M. Tůma. 2023. *Algorithms for Sparse Linear Systems* (first ed.). Birkhäuser, Cham.

[37] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B. Khaled Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 1619–1629. https://doi.org/10.1145/3459637.3482264

[38] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I. Nikolenko. 2020. RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 528–536. https://doi.org/10.1145/3336191.3371831

[39] Barry F. Smith, Petter E. Bjorstad, and William D. Gropp. 1996. *Domain decomposition.* Cambridge University Press, Cambridge, UK. xii+224 pages. Parallel multilevel methods for elliptic partial differential equations.

[40] Harald Steck. 2019. Collaborative Filtering via High-Dimensional Regression. arXiv:1904.13033 [cs.IR]

[41] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3251–3257. https://doi.org/10.1145/3308558.3313710

[42] Harald Steck. 2019. Markov Random Fields for Collaborative Filtering. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, Article 491, 12 pages.

[43] Harald Steck and Dawen Liang. 2021. Negative Interactions for Improved Collaborative Filtering: Don't Go Deeper, Go Higher. In *Proceedings of the 15th ACM Conference on Recommender Systems* (Amsterdam, Netherlands) *(RecSys '21)*. Association for Computing Machinery, New York, NY, USA, 34–43. https://doi.org/10.1145/3460231.3474273

[44] Jan Van Balen and Bart Goethals. 2021. High-Dimensional Sparse Embeddings for Collaborative Filtering. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 575–581. https://doi.org/10.1145/3442381.3450054

[45] Arno C. N. van Duin. 1999. Scalable parallel preconditioning with the sparse approximate inverse of triangular matrices. *SIAM J. Matrix Anal. Appl.* 20, 4 (1999), 987–1006. https://doi.org/10.1137/S0895479897317788

[46] Vojtěch Vančura, Rodrigo Alves, Petr Kasalický, and Pavel Kordík. 2022. Scalable Linear Shallow Autoencoder for Collaborative Filtering. In *Proceedings of the 16th ACM Conference on Recommender Systems* (Seattle, WA, USA) *(RecSys '22)*. Association for Computing Machinery, New York, NY, USA, 604–609. https://doi.org/10.1145/3523227.3551482

[47] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/3331184.3331267

[48] Zygmunt Zajac. 2017. Goodbooks-10k: a new dataset for book recommendations. http://fastml.com/goodbooks-10k.