Generative modeling of Sparse Approximate Inverse Preconditioners

Mou Li[1][0009−0009−0216−9292], He Wang[2][0000−0002−2281−5679], and Peter K. Jimack[3][0000−0001−9463−7595]

[1] University of Leeds, Leeds LS2 9JT, UK
scmli@leeds.ac.uk
[2] University College London, London WC1E 6BT, UK
he_wang@ucl.ac.uk
http://drhewang.com
[3] University of Leeds, Leeds LS2 9JT, UK
p.k.jimack@leeds.ac.uk

Abstract. We present a new deep learning paradigm for the generation of sparse approximate inverse (SPAI) preconditioners for matrix systems arising from the mesh-based discretization of elliptic differential operators. Our approach is based upon the observation that matrices generated in this manner are not arbitrary, but inherit properties from differential operators that they discretize. Consequently, we seek to represent a learnable distribution of high-performance preconditioners from a low-dimensional subspace through a carefully-designed autoencoder, which is able to generate SPAI preconditioners for these systems. The concept has been implemented on a variety of finite element discretizations of second- and fourth-order elliptic partial differential equations with highly promising results.

Keywords: Deep learning · Sparse matrices · Preconditioning · Elliptic partial differential equations · Finite element methods.

1 Introduction

Finding the solution of systems of linear algebraic equations,

$$Ax = b\,, \tag{1}$$

has been a core topic in the field of scientific computation for many decades. Such systems arise naturally in a wide range of algorithms and applications, including from the discretization of systems of partial differential equations (PDEs). Not only are the equations (1) the main product of the discretization process for linear PDEs but such systems must be solved during every nonlinear iteration when solving nonlinear PDE systems [4]. Furthermore, this step is often the most computationally expensive in any given numerical simulation.

Direct methods, based on the factorisation of the matrix A into easily invertible triangular matrices are designed to be highly robust. However they generally

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

scale as $O(n^3)$ as the system size, n, increases. Hence as n approaches values in the millions the computational cost and memory requirements quickly become unacceptable. Fortunately, when the matrix A is sparse, it is possible to optimize the factorization process to minimize the fill-in and avoid storing (or computing with) the zero entries in A [21]. Such sparse direct methods significantly reduce the computational complexity but can still suffer from unacceptable memory and CPU requirements when n becomes sufficiently large. Consequently iterative solution methods are a popular choice for large sparse systems (1).

Many iterative methods have been developed as an alternative approach to direct methods, offering a flexible level of accuracy as the trade-off for memory storage and computational cost. The Conjugate Gradient (CG) algorithm is arguably the most well-developed and efficient iterative method for solving large, sparse, symmetric positive definite (SPD) systems, whilst other Krylov-subspace algorithms are widely-used for indefinite or non-symmetric systems [11]. In the SPD case, the convergence of the CG method is dependent on the distribution of the eigenvalues of the stiffness matrix A [33]. Assuming that these eigenvalues are widely spread along the real line, the CG algorithm will suffer from slow convergence when the problem is ill-conditioned. That is, the condition number $\kappa(A) \gg 1$, which is the ratio of the maximum, $\lambda_{\max}(A)$, over the minimum, $\lambda_{\min}(A)$, eigenvalues of A. Consequently, a longstanding topic of research has been the development of preconditioning techniques to reduce $\kappa(A)$ (strictly $\kappa(AP)$) via a transformation of the stiffness matrix A through multiplication with a carefully selected non-singular matrix P on both sides of the equation: $APx = bP$.

Selection of a "good" preconditioner, P should take account of the following two requirements:

1. It should be easy to construct and cheap to implement (i.e. to solve $Px = y$ for a given vector y).
2. It should significantly reduce the condition number of the modified problem, $\kappa(AP)$.

In general these requirements are in conflict with each other, so frequently we seek to use prior knowledge about the linear system in order to manage the trade-off between point one and two. If little is known about A then one of the simplest choices for P is the diagonal of A (frequently referred to as Jacobi preconditioning [11]), which clearly satisfies the first requirement above but typically satisfies the second requirement less well. More complex preconditioners, which aim to do better on the second requirement, include incomplete factorization approaches such as incomplete Cholesky (IC) and incomplete LU (ILU) decompositions [7, 11], multigrid (MG) and algebraic multigrid (AMG) approaches [5, 15], domain decomposition (DD) methods [27], and sparse approximate inverse (SPAI) methods [2] which are the main topic of this paper. In recent developments there are also some attempts to utilise data-driven methods to generate preconditioners [3, 17, 20, 23, 24, 31].

If the exact inverse of A were known then setting $P = A^{-1}$ would perfectly satisfy condition 2 above. However, the cost of constructing and implementing

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

this P would mean that condition 1 would fail to be satisfied. Sparse approximate inverse (SPAI) methods seek to balance these requirements more evenly by constructing a cheaper approximation to A^{-1} ($P \approx A^{-1}$), where P is a sparse matrix so as to ensure that its application, $y = Px$, has a cost of $O(n)$ rather than $O(n^2)$. Designing a high performance preconditioner by the conventional numerical approaches mentioned above requires prior knowledge and past experience of specific families of PDE systems. The final product also varies with chosen parameters, this suggests that there is a distribution of high-performance preconditioners for A, which naturally justifies using generative models. In this paper, we propose a deep learning based generative model for constructing a SPAI preconditioner, P, for a given SPD matrix A arising from the finite element discretization of a self-adjoint PDE or system. Our method uses a conditioned variational auto-encoder to map the conditioned distribution of $p(A^{-1}|A)$ into a lower dimensional latent space. After training, it is able to generate high-performance preconditioners for SPD matrices arising from the discretization of unseen self-adjoint problems under the same mesh density.

Whilst this is not the first research to propose the use of machine learning (ML) techniques to generate SPAI preconditioners (see [31] for example), we believe that this is the first to consider the use of a graph representation of the sparse matrix A and the first to consider modeling the distribution of A and A^{-1} for such a task. This permits the representation of A in a smaller latent space through our use of a variational auto-encoder. Consequently, since our model only cares about the latent data distribution of the inverse of the stiffness matrix A, it is easily generalisable to other self-adjoint problems. Furthermore, we propose a controllable sparsity pattern for the preconditioner to allow a trade-off between the performance and the computational cost.

2 Related Work

In this section we provide a brief overview of the most common approaches to preconditioning that are in widespread use in numerical codes today. This is then followed by a short subsection on existing research into the use of data-driven methods for generating preconditioners. As part of that section we also highlight some of the proposed techniques for using ML to solve PDE systems since these could also be used to motivate new preconditioners (i.e. by applying as a preconditioner rather than a solver).

2.1 Numerical preconditioning

Researchers have studied different methods for generating effective preconditioners over many decades, leading to a vast body of work on this topic [34]. For the purposes of this paper we restrict our attention to sparse SPD matrices, A, arising from mesh-based discretizatons of self-adjoint PDEs. As already noted, the simplest approach that is in widespread use is Jacobi preconditioning, where $P = \text{diag}(A)^{-1}$. This is easy to construct and cheap to implement,

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

but not sufficiently effective for many important problems [23]. Consequently more sophisticated techniques are frequently required, such as those based upon incomplete factorizations of A. Incomplete Cholesky (IC) factorization is most appropriate for SPD matrices, where $A \approx LL^T$. The sparsity pattern of the lower triangular matrix L can be chosen to be identical to that of the lower triangle of A or can be slightly more generous based upon a drop tolerance that is used to decide whether to allow any "fill-in" during the factorization process [34]. Note that this technique requires a forward and a backward substitution in order to apply the preconditioner, which is not generally well-suited to efficient parallel implementation. Furthermore, this approach requires careful design to balance the computational cost and accuracy associated with different levels of fill-in (complete fill-in leads to perfect factorization, which means that $\kappa(AP) = 1$ but at prohibitive cost, whereas insufficient fill-in can lead to a sub-optimal condition number).

The other class of preconditioners that we discuss in detail here are sparse approximate inverse (SPAI) methods, which are the main topic of this paper. This variant evaluates the preconditioner P to be a sparse approximation to the inverse of the stiffness matrix A. Frobenius norm minimisation and incomplete bi-conjugation are the two most widely implemented frameworks to compute the SPAI preconditioners [4, 34]. One major difficulty of SPAI preconditioning is the choice of the sparsity pattern. Since the inverse of an irreducible sparse matrix is proven to be a structurally dense matrix [9], when the sparsity pattern is predefined, such preconditioners may not work well if there exist entries with large magnitude outside the defined pattern. Attempts to address this by automatically capturing a pseudo-optimal sparsity pattern include the use of a drop tolerance [8] or of a “profitability factor” via a residual reduction process [13, 16]. In this work we consider only predefined sparsity patterns based around the sparsity of the family of matrices, A, being considered.

2.2 Data driven preconditioning

In recent years a number of approaches have been proposed for the direct solution of systems of PDEs using both supervised and unsupervised ML methods. Noteworthy, and highly influential, early examples include the deep Ritz method [10], the deep Galerkin method [32] and physics-informed neural networks (PINNs) [28–30]. The latter approach has led to a substantial, and rapidly increasing, body of research into the unsupervised learning of PDE solutions and related inverse problems. However, as forward solvers, PINNs are not generally competitive with classical numerical approaches based upon efficient preconditioning. It is for this reason, and inspired by the successes of PINNs, that we seek to utilise the power of machine learning to generate high-performance preconditioners (as opposed to complete solvers).

There is relatively little prior work on the use of machine learning to develop preconditioners for use within conventional numerical algorithms. The first attempt to generate a SPAI preconditioner appears in [31], where a convolutional

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

neural network (CNN) is used to derive an approximate triangular factorization of the inverse matrix. More recent research, such as [17, 23] has focused on incomplete LU and Cholesky preconditioners, with the former targeting the non-self-adjoint Navier-Stokes equations and the latter two considering SPD systems: the work of [23] being most similar to that considered in this paper due to their use of a GNN. Other approaches that have been taken to develop novel preconditioners include [3], which mimics a multigrid approach through a combination of a CNN-based smoother and coarse-grid solvers, and [20], which builds a preconditioner directly upon a DeepONet approximation of the operator that represents the solution of the underlying PDE [24].

3 Methodology

In this initial investigation we focus on distributions of sparse matrices, A, arising from the finite element discretization of elliptic PDEs: the precise sparsity pattern depending upon the differential operator, the finite element spaces used, and the mesh on which the discretization occurs. Even though the inverse matrices, $A^{-1} \in \mathbb{R}^{n\times n}$ are generally dense in structure, other preconditioning methods are able to approximate cheaper versions of A^{-1}, which led us to assume that there exists a learnable distribution of high performance preconditioners within a lower dimensional subspace of $\mathbb{R}^{n\times n}$ with a prescribed sparsity pattern. We then propose a graph-conditioned variational autoencoder (GCVAE) architecture which is able to generate such SPAI preconditioners for these SPD linear systems.

3.1 Problem setup

Inspired by the traditional Frobenius norm minimization approach of SPAI preconditioning, our starting point is the following assumption:

$$\forall A \in S_A \quad \exists R \in S_M : \left\| I - R^T A R \right\|_F \approx 0 \,, \tag{2}$$

where S_A is our set of possible sparse SPD matrices and $S_M \subset \mathbb{R}^{n\times n}$ with a prescribed sparsity pattern defined by a selected mask, M. Let us consider the dataset $\mathcal{A} = \{A^{(i)}\}_{i=1}^{N}$ having N independently and identically distributed samples and drawn from an unknown distribution $p(\mathcal{A})$, generated from a given family of linear PDE problems. Let $\mathcal{A}^{-1} = g(\mathcal{A})$ where $g(\cdot)$ is the inverse matrix transformation function. We assume that the conditional distribution of $p(\mathcal{A}^{-1}|\mathcal{A})$ lies within a parametric family of distributions, such as Gaussian, on a lower-dimensional latent space such that $p_\theta(z|\mathcal{A}, \mathcal{A}^{-1}) = \mathcal{N}(\mu, \sigma^2)$, where θ represents the learnable parameters of the neural networks. We also assume that for every given $A^{(i)}$ there exists a set of high-performance preconditioners $\mathcal{R}^{(i)}$ which satisfies eq. 2 with a given sparsity pattern M and $\mathcal{R} \sim p_\theta(z|\mathcal{A}, \mathcal{A}^{-1})$. Then the SPAI preconditioners $\mathcal{R}$ can be generated from the generative distribution $p_\theta(\mathcal{R}|z, \mathcal{A})$ conditioned on input $\mathcal{A}$, the prescribed sparsity pattern of $\mathcal{R}$, and latent variable $z \sim \mathcal{N}(\mu, \sigma^2)$.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

The application of variational auto-encoders (VAEs) has shown a great potential in learning the probability distribution of a given dataset [18] and, when coupled with a condition, can ensure the robustness of the generative model [26]. VAEs have also shown great potential in modeling the distribution of the utility matrix for a recommender system [1] and for a gene expression matrix [25], both based upon large and sparse matrices. Therefore, we propose a conditional VAE generative model to approximate SPAI preconditioners.

3.2 Architecture design

As noted above, the type of architecture we now consider is a conditioned variational auto-encoder (cVAE) [14]. The encoder part consists of two different encoders, which we now discuss in turn.

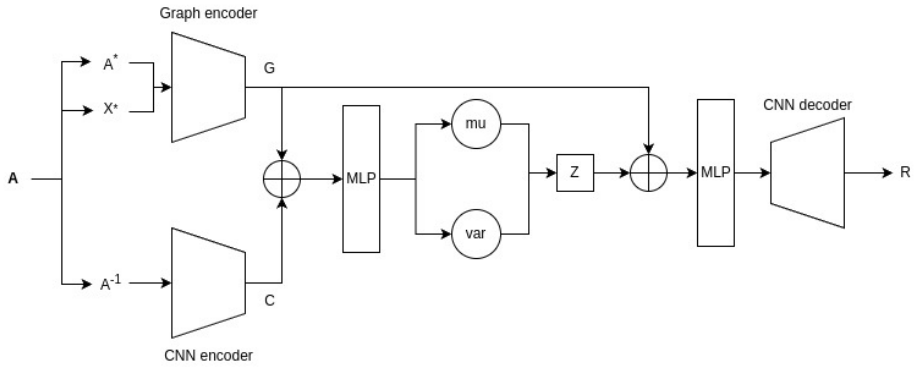

Fig. 1. Schematic diagram of the proposed graph-conditioned variational autoencoder

As shown in Figure 1, the first is a conditional part (a graph encoder) that consists of several layers of a graph neural network (GNN). The input data A is a sparse matrix which is represented as a graph, where the diagonal entries and off-diagonal entries are the nodes and edges of the graph respectively. GNNs have two clear advantages over CNNs under our setting: 1) a GNN can perform the feature mapping with a full "view" of the input data through its message passing mechanism, 2) a GNN is only interested in the entries that are non-zero therefore it is a sparse implementation, which significantly reduces the memory requirement and the training cost. The graph VAE is first proposed by [19], where it is used to learn latent representations of unweighted undirected graphs with multidimensional node features for link prediction in citation networks. In our case, the system matrix A can be treated as a weighted and directed graph with single dimensional node features. Therefore we replace the graph convolutional layer (GCN) with the graph attention layer (GATv2) [6]. This graph encoder

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

takes A^* and X^* as the input, where A^* is the self-looped adjacency matrix of A and X^* is the node feature matrix (where $X^* \in \mathbb{R}^{n\times 1}$ and n is the number of nodes).

The second encoder is chosen to be a CNN encoder since it takes A^{-1}, the exact inverse of the system matrix A, as the input. CNNs are known to be powerful and efficient feature extractors for image data [12], where grey-scale images are fundamentally dense matrices. During training, we assume the latent distribution is a Gaussian, i.e. $z \sim \mathcal{N}(\mu, \sigma^2)$, both encoders' outputs are concatenated and then passed through a multilayer perceptron (MLP) to approximate the mean(μ) and the log variance(ϕ) of the latent distribution. The latent variable z is then reconstructed deterministically through the reparameterization trick [18] i.e. $z = \mu + \sigma\epsilon$, where $\sigma = e^{\frac{1}{2}\phi}$, ϵ is an auxiliary noise variable sampled from standard Gaussian such that $\epsilon \sim \mathcal{N}(0, 1)$. This technique makes the stochastic estimation of the latent variable z differentiable. The decoder reconstructs $R \sim A^{-1}$ from z such that $\left\|I - R^T AR\right\|_F \approx 0$. We implemented the same optimisation function as the traditional SPAI algorithms, because the eigenvalue decomposition operation is not differentiable and this least-square optimisation implicitly reduces the condition number of AR which is what we are interested in, as the condition number of an identity matrix and its multiplications with an arbitrary scalar is 1. The sparsity pattern of R is constrained by a mask M which has a similar sparsity pattern of A at the output layer. Without the prior knowledge of A^{-1}, this is the simplest method of predefining the sparsity pattern as stated in [34]. Furthermore, we allow a small percentage of extra non-zeros in addition to the existing non-zero entries, which we have found to significantly improve the model performance in our experiments.

For inference, as shown in Figure 2 we pass the adjacency matrix A^* and the node feature X^* of an unseen stiffness matrix A from the same family of linear PDE problems through the graph encoder to generate the conditional information G, then concatenate with z sampled from $z \sim \mathcal{N}(\mu, \sigma^2)$ and pass it through the decoder to generate the preconditioner R. Due to the nature of our model, multiple different R can be generated for a single A.

3.3 Loss function

In general, the family of VAE networks optimise the well-known evidence lower bound (ELBO) [18]:

$$L = \mathbb{E}[\log p(R|Z)] - \alpha D_{KL}[q(Z|X)||p(Z)], \tag{3}$$

where the prior $Z \sim \mathcal{N}(0, I)$ s.t. $p(Z) = \prod p(z_i) = \prod \mathcal{N}(z_i|0, I)$. This type of loss function contains a reconstruction term, which is the first term that calculates the expected negative reconstruction error. The second term is the KL-divergence term, and it can be regarded as a regulariser which encourages the posterior distribution to be close to the prior, in this case the prior is assumed as the standard Gaussian. In our case, we replace the reconstruction term with $\left\|I - R^T AR\right\|_F$ and minimise the following loss function:

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

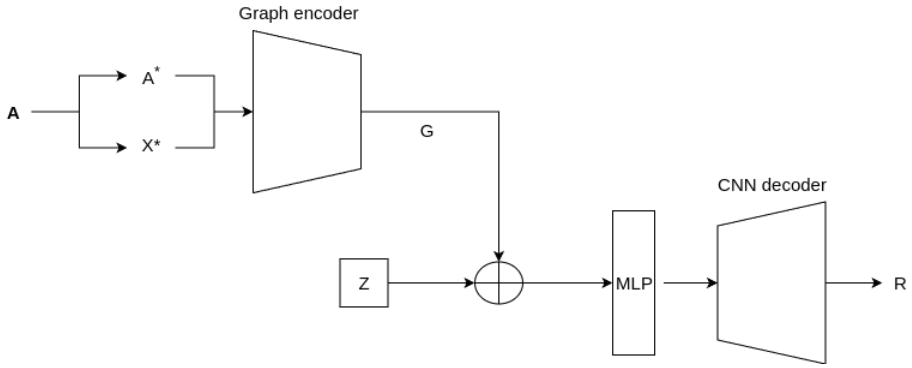

Fig. 2. Schematic diagram of the inference process

$$L = \mathbb{E}[\left\|I - R^T AR\right\|_F^2] - \alpha D_{KL}[q(Z|X)||p(Z)]$$
$$\textit{for} \quad X = GNN_\theta(A^*, X^*) \oplus CNN_\theta(A^{-1}) \,, \tag{4}$$

where α is the regularising parameter, which is chosen to be 0.1. $GNN_\theta(\cdot)$ and $CNN_\theta(\cdot)$ represent the GNN encoder and CNN encoder as shown in Figure 1, and $\oplus$ is the aggregation operator.

4 Experiments

In the first set of computational experiments that we consider (subsections 4.1 and 4.2) the dataset, $\mathcal{A}$, is drawn from piecewise linear finite element discretizations of a family of second order elliptic PDEs in two dimensions. The second set of experiments considers a more challenging dataset, $\mathcal{A}$, that is drawn from discontinuous (piecewise quadratic) Galerkin discretizations of a family of two-dimensional biharmonic problems (subsections 4.3 and 4.4). Such problems are known to lead to highly ill-conditioned matrix systems upon discretization (as illustrated below). Both experiments are trained on a single NVDIA RTX 3090 GPU with 24 Gigabyte of VRAM. The dataset preparation of both problems takes less than 30 mins for the largest problem size. For the largest problem size of 2D Poisson's problem 1873×1873, the training converges at 200 epochs and around 6 mins per epoch. Similarly, for the largest biharmonic problem, of size 1089×1089, the training converges at 260 epochs and around 2 mins per epoch. The training cost is expected to grow linearly with the total number of hyper-parameters, which depends on many different factors, such as the discretized mesh density, parallelisation, depth of the model, number of the channels of the CNN layers, etc. For practical implementation of the model, fine tuning of the hyperparameters is required to find the optimal balance between the training cost and model performance, though this is not the focus of this paper.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

4.1 2D Poisson's Problem

We generate sets of $N = 2000$ matrices based upon the piecewise linear finite element discretization of PDEs of the form

$$\underline{\nabla} \cdot (f(\underline{x})\underline{\nabla} u) = g(\underline{x}), \tag{5}$$

in two dimensions. For simplicity we consider a unit square domain and impose Dirichlet conditions on the entire boundary. The dimension, n, of the resulting stiffness matrices, A, is equal to the number of interior node points in the unstructured triangular mesh, whilst the entries of A depend upon the node locations and the choice of $f(\underline{x})$, which is drawn from a family of polynomial functions that are positive on the unit square. Each set of 2000 matrices is generated on the same mesh and is randomly split into 1600 training samples and 400 for testing.

4.2 Results and discussion - Poisson family

This section demonstrates the performance of the preconditioners generated by our GCVAE model when applied with a CG solver using a relative convergence criterion of 1.0e-5. Included within our results is a comparison with two baseline methods: Jacobi preconditioning and Super Nodal incomplete LU factorisation (SPILU) [21] (using its symmetric mode to obtain IC preconditioning). Note that the performance of the latter comparator depends critically on the value of a "drop tolerance" parameter, which controls the amount of fill-in that is permitted during the incomplete factorization of A. When this is very small the IC preconditioner is highly effective but at the expense of significant additional computational cost (reducing the overall efficiency of the solver); when the drop-tolerance is larger the cost of computing and applying the preconditioner goes down but at the expense of a much larger number of CG iterations.

Figure 3 (left) shows the average estimated condition numbers for preconditioned systems with each choice of preconditioner. For the SPILU case we have artificially selected the drop-tolerance for each problem size so as to match the number of iterations taken using the GCVAE preconditioner. Consequently, by design, the equivalent curve in Figure 3 (right), which shows average CG iterations, completely overlays the GCVAE curve. Also in this graph are the average iteration counts for SPILU applied in its more conventional form, with a constant drop-tolerance (in this case 0.12). It is clear from these examples that both the condition number and the CG iteration counts with Jacobi preconditioning grow at a much faster rate than for the GCVAE approach, demonstrating that our model will out-perform the Jacobi method on total execution time for sufficiently large problems.

Comparison against SPILU is less straightforward due to the trade-offs in the choice of drop-tolerance described above. The SPILU algorithm uses $(A^T + A)$ based column permutation [22] which causes the condition number to increase rapidly with the problem size as shown in Figure 3 (left). This illustrates the

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

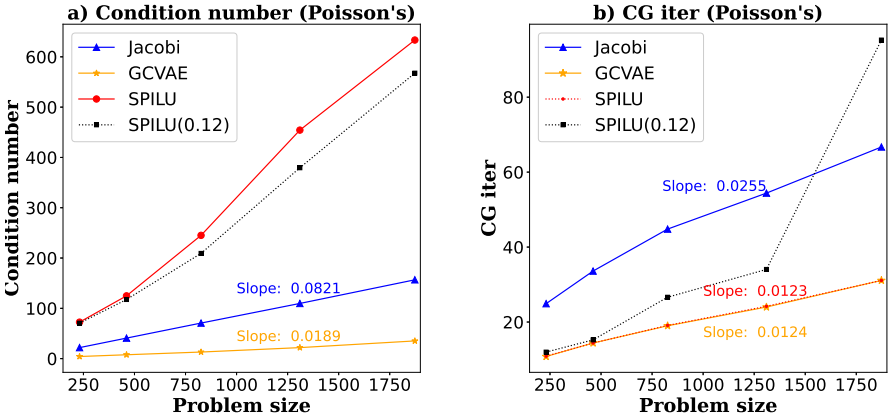

Fig. 3. Comparison of the condition number and CG algorithm iteration count for Jacobi preconditioning, SPILU preconditioning and our proposed method (GCVAE) when applied to discretizations of a family of second order operators

limitation of using the condition number as the measure of quality, which is why we prefer to use iteration count. Nevertheless, even with this measure, Figure 3 (right) shows that the parameters of the SPILU method need to be carefully tuned to achieve optimal performance. By carefully reducing drop-tolerance as n increases we are able to match the iteration counts of the GCVAE preconditioner, though the latter has significantly lower condition numbers. In Figure 5 (left) we also compare the density of non-zero entries in these two preconditioners for different choices of n, in order to give an indication of their computational costs. It can be seen that for sufficiently large systems the GCVAE preconditioner will be expected to have fewer non-zeros. Furthermore, to apply the SPILU preconditioner requires backward and forward substitution to be applied which cannot naturally be done in parallel, whereas the SPAI preconditioning is ideally suited to parallel implementation (since it just requires a sparse matrix-vector multiplication). The execution time benchmark test has not been carried out as our model is currently implemented in a non-optimised manner in a Python environment, whereas the state-of-art numerical software SPILU is implemented in a highly optimised C-language environment. Nevertheless, the advantage of the data driven approach is that, once it is tuned and trained, it can be used as a black box tool with execution time complexity close to $\mathcal{O}(n)$.

4.3 Biharmonic Problem

We again generate sets of $N = 2000$ matrices, this time based upon the piecewise quadratic discontinuous Galerkin discretization of fourth order PDEs of the form

$$\nabla^2(f(\underline{x})\nabla^2 u) = g(\underline{x}) \tag{6}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

in two dimensions. We consider a unit square domain and impose Dirichlet conditions on both u and $\nabla^2 u$ on the entire boundary. For a given triangular mesh, the dimension, n, of the resulting stiffness matrices, A, is much greater than for the piecewise linear approximations previously considered and the condition number of the matrix A is much larger (hence this is a considerably more challenging test problem). The individual non-zero entries of A depend upon the mesh node locations and the choice of $f(\underline{x})$, which is again drawn from a family of polynomial functions that are positive on the unit square. As previously, each set of 2000 matrices is generated on the same mesh and is split into 1600 training samples and 400 for testing. For this problem, we allow 20% extra number of non-zeros upon the existing non-zero entries during training and designed 4 CNN layers for its encoder and decoder to guarantee the convergence and model performance. In contrast, we only used 3 CNN layers with less channels per layer for 2D Poisson's problem.

4.4 Results and discussion - Biharmonic family

Similarly, in this subsection, we compare our method against two baseline methods: Jacobi and SPILU. As shown in Figure 4, for this ill-conditioned problem, as the problem size increases our method significantly outperforms Jacobi preconditioning in terms of both the condition number reduction and the CG convergence rate.

Comparison against SPILU shows similar features as for the previous test case. For the reasons described previously we do not find condition number to be a useful metric in this case and therefore focus on iteration counts. Figure 4 (right) shows that it is possible to tune the value of drop-tolerance for each problem size in order to match the number of CG iterations obtained with the GCVAE preconditioner, as well as showing the growth in iterations when a constant drop-tolerance is used (in this case 1.2×10^{-4}). Equally significantly however, we see from Figure 5 (right) that the number of non-zeros required for the GCVAE preconditioner is significantly fewer than for the SPILU preconditioner, even when drop-tolerance is tuned to match the number of iterations. Combined with the fact that application of SPILU requires backward and forward substitution, we observe that this preconditioner is much more expensive to apply than our proposed approach.

5 Conclusions and Future Work

5.1 Conclusions

This paper proposes a novel generative modelling framework to generate sparse approximate inverse preconditioners for matrix systems arising from the mesh-based discretization of families of self-adjoint elliptic partial differential equations. The approach has shown excellent potential in terms of its ability to reduce the condition number of the systems considered and to increase the convergence

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

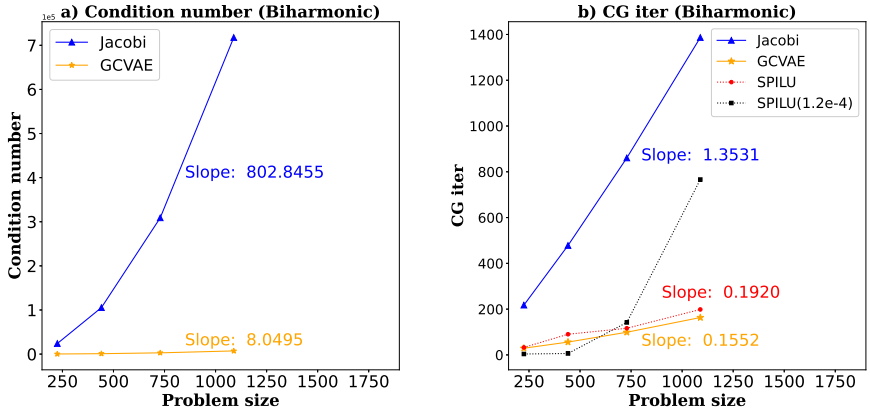

Fig. 4. Comparison of the condition number (in the unit of 1.0×10^5) and CG algorithm iteration count for Jacobi preconditioning, SPILU preconditioning and our proposed method (GCVAE) when applied to discretizations of a family of fourth order operators. The condition numbers of the SPILU approaches are not illustrated in **a)** as they are too large to fit into the plot.

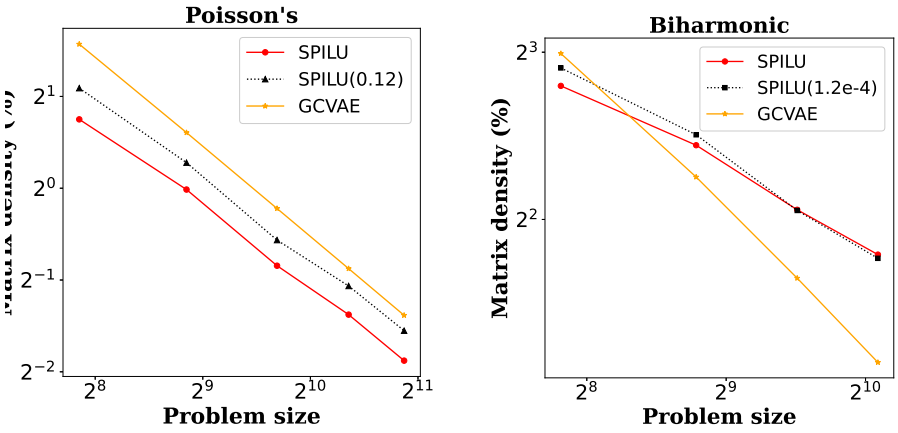

Fig. 5. Comparison of the proportion of non-zero entries in the preconditioners generated via the SPILU and GCVAE methods for the second order (left) and fourth order (right) problems

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

rate of the conjugate gradient solver. This performance has been demonstrated for two different families of partial differential equations: of second and fourth order respectively. Furthermore, comparing against a state-of-art factorisation-based preconditioning technique we observe that, for sufficiently large systems, our proposed approach should require fewer floating point operations to apply and that it is better suited to future parallel implementation.

5.2 Limitation and Future Work

Based upon the evidence presented here we believe that the proposed GCVAE approach has significant potential. However, there still exists several limitations which will need to be considered for future development. The most significant of these is that our training phase currently requires the inverse matrices A^{-1} to be known for our training set, even though the dataset preparation cost is negligible compared with the training time for the experimented problem size, this cost will quickly grow to an unaccepted level when generalising to solve real world problems. Consequently, we propose to investigate ways to relax this restriction (e.g. through the use of training data from coarser finite element grids or the application of techniques from algebraic multigrid to obtain coarser representations of training matrices and inverting these). Furthermore, despite our current masking scheme allowing flexible mask selection that can be tuned to deal with more complex PDE problems, the sparsity pattern does need to be decided prior to training which may restrict the generalisation capability of the model. A learning-based masking approach could be more flexible and efficient, so will be attempted in further development. Finally, the work in this paper is limited to self-adjoint elliptic problem, which result in symmetric positive-definite stiffness matrices. We also intend to generalise our model to other families of PDE problems, which lead to non-symmetric and/or indefinite linear systems.

Acknowledgments. The first author gratefully acknowledges receipt of a doctoral training award from the UK Engineering and Physical Sciences Research Council.

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

References

1. Ali, A., Yu, L., Farida, M., Majjed, A.Q.: Variational autoencoder Bayesian matrix factorization (VABMF) for collaborative filtering. Applied Intelligence **51**(7), 5132–5145 (Jul 2021). https://doi.org/10.1007/s10489-020-02049-9
2. Anzt, H., Huckle, T.K., Bräckle, J., Dongarra, J.: Incomplete sparse approximate inverses for parallel preconditioning. Parallel Computing **71**, 1–22 (2018)
3. Azulay, Y., Treister, E.: Multigrid-augmented deep learning preconditioners for the Helmholtz equation. SIAM Journal on Scientific Computing **45**(3), S127–S151 (2023). https://doi.org/10.1137/21M1433514

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

4. Benzi, M.: Preconditioning Techniques for Large Linear Systems: A Survey. Journal of Computational Physics **182**(2), 418–477 (Nov 2002). https://doi.org/10.1006/jcph.2002.7176
5. Bramble, J.H.J.H.: Multigrid methods. Pitman research notes in mathematics series, 294, Longman Scientific and Technical, Harlow (1993)
6. Brody, S., Alon, U., Yahav, E.: How Attentive are Graph Attention Networks? arXiv:2105.14491v3 [cs.LG] (Jan 2022). https://doi.org/10.48550/arXiv.2105.14491
7. Chan, T.F., Van der Vorst, H.A.: Approximate and Incomplete Factorizations, pp. 167–202. Springer Netherlands, Dordrecht (1997). https://doi.org/10.1007/978-94-011-5412-3_6
8. Chow, E., Saad, Y.: Approximate Inverse Preconditioners via Sparse-Sparse Iterations. SIAM Journal on Scientific Computing **19**(3), 29 (May 1998). https://doi.org/10.1137/S1064827594270415
9. Duff, I.S., Erisman, A.M., Gear, C.W., Reid, J.K.: Sparsity structure and Gaussian elimination. ACM SIGNUM Newsletter **23**(2), 2–8 (Apr 1988). https://doi.org/10.1145/47917.47918
10. E, W., Yu, B.: The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. Communications in Mathematics and Statistics **6**(1), 1–12 (Mar 2018). https://doi.org/10.1007/s40304-018-0127-z
11. Golub, G.H.G.H., Van Loan, C.F.: Matrix computations. Johns Hopkins studies in the mathematical sciences, The Johns Hopkins University Press, Baltimore, fourth edition. edn. (2013)
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cumberland, United States (2016)
13. Grote, M.J., Huckle, T.: Parallel Preconditioning with Sparse Approximate Inverses. SIAM Journal on Scientific Computing **18**(3), 16 (May 1997). https://doi.org/10.1137/S1064827594276552
14. Gu, C., Zhao, S., Zhang, C.: Diversity-Promoting Human Motion Interpolation via Conditional Variational Auto-Encoder. arXiv:2111.06762v1 [cs.CV] (Nov 2021), http://arxiv.org/abs/2111.06762
15. Henson, V.E., Yang, U.M.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics **41**(1), 155–177 (2002). https://doi.org/https://doi.org/10.1016/S0168-9274(01)00115-5
16. Huckle, T.: Factorized Sparse Approximate Inverses for Preconditioning. The Journal of Supercomputing **25**(2), 109–117 (Jun 2003). https://doi.org/10.1023/A:1023988426844
17. Häusner, P., Öktem, O., Sjölund, J.: Neural Incomplete Factorization: Learning Preconditioners for the Conjugate Gradient Method. arXiv:2305.16368v2 [math.OC] (Feb 2024). https://doi.org/10.48550/arXiv.2305.16368
18. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. arXiv:1312.6114 [cs, stat] (May 2014), http://arxiv.org/abs/1312.6114
19. Kipf, T.N., Welling, M.: Variational Graph Auto-Encoders. arXiv:1611.07308 [cs, stat] (Nov 2016), http://arxiv.org/abs/1611.07308
20. Kopaničáková, A., Karniadakis, G.E.: DeepOnet Based Preconditioning Strategies For Solving Parametric Linear Systems of Equations. arXiv:2401.02016v2 [math.NA] (Jan 2024). https://doi.org/10.48550/arXiv.2401.02016
21. Li, X.S.: An overview of SuperLU: Algorithms, implementation, and user interface. ACM Trans. Math. Software **31**(3), 302–325 (September 2005)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40

22. Li, X., Demmel, J., Gilbert, J., iL. Grigori, Shao, M., Yamazaki, I.: SuperLU Users' Guide. Tech. Rep. LBNL-44289, Lawrence Berkeley National Laboratory (September 1999), https://portal.nersc.gov/project/sparse/superlu/ug.pdf Last update: June 2018
23. Li, Y., Chen, P.Y., Du, T., Matusik, W.: Learning Preconditioner for Conjugate Gradient PDE Solvers. arXiv:2305.16432v2 [math.NA] (Sep 2023). https://doi.org/10.48550/arXiv.2305.16432
24. Lu, L., Jin, P., Karniadakis, G.E.: DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. Nature Machine Intelligence **3**(3), 218–229 (Mar 2021). https://doi.org/10.1038/s42256-021-00302-5
25. Lukassen, S., Ten, F.W., Adam, L., Eils, R., Conrad, C.: Gene set inference from single-cell sequencing data using a hybrid of matrix factorization and variational autoencoders. Nature Machine Intelligence **2**(12), 800–809 (Dec 2020). https://doi.org/10.1038/s42256-020-00269-9
26. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. arXiv:1411.1784v1 [cs.LG] (Nov 2014), https://arxiv.org/abs/1411.1784v1
27. Quarteroni, A., Valli, A.A.: Domain decomposition methods for partial differential equations. Numerical mathematics and scientific computation, Clarendon Press, Oxford (1999)
28. Raissi, M., Perdikaris, P., Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686–707 (2019)
29. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. arXiv:1711.10561 [cs, math, stat] (Nov 2017), http://arxiv.org/abs/1711.10561
30. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. arXiv:1711.10566 [cs, math, stat] (Nov 2017), http://arxiv.org/abs/1711.10566
31. Sappl, J., Seiler, L., Harders, M., Rauch, W.: Deep Learning of Preconditioners for Conjugate Gradient Solvers in Urban Water Related Problems. arXiv:1906.06925 [cs, math, stat] (Jun 2019), http://arxiv.org/abs/1906.06925
32. Sirignano, J., Spiliopoulos, K.: DGM: A deep learning algorithm for solving partial differential equations. Journal of Computational Physics **375**, 1339–1364 (Dec 2018)
33. Tadmor, E.: A review of numerical methods for nonlinear partial differential equations. Bulletin (New Series) of the American Mathematical Society **49** (Oct 2012). https://doi.org/10.1090/S0273-0979-2012-01379-4
34. Wathen, A.J.: Preconditioning. Acta Numerica **24**, 329–376 (May 2015)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_40