V. Grimm, A. Heinlein, A. Klawonn

A short note on solving partial differential equations using convolutional neural networks

Technical Report ID: CDS-2022-07
Available at `https://nbn-resolving.org/urn:nbn:de:hbz:38-642271`
Submitted on November 29, 2022

A short note on solving partial differential equations using convolutional neural networks

Viktor Grimm, Alexander Heinlein, and Axel Klawonn

1 Introduction

Solving partial differential equations (PDEs) is a common task in numerical mathematics and scientific computing. Typical discretization schemes, for example, finite element (FE), finite volume (FV), or finite difference (FD) methods, have the disadvantage that the computations have to be repeated once the boundary conditions (BCs) or the geometry change slightly; typical examples requiring the solution of many similar problems are time-dependent and inverse problems or uncertainty quantification. Every single computation, however, can be very time consuming, motivating the development of surrogate models that can be evaluated quickly. There exist some possible surrogate models, including reduced order models [8], reduced basis, and neural network-based models.

In this work, we will discuss an approach for predicting the solution of boundary value problems using convolutional neural networks (CNNs). This approach is particularly interesting in the context of surrogate models which predict the solution based on a parametrization of the model problem, for instance, with respect to variations in the geometry or BCs; cf. fig. 1 for a sketch of the CNN-based surrogate modeling approach. In [6, 2, 3], a CNN model has been trained to predict stationary flow inside a channel with an obstacle of varying geometry; the model is trained in a purely data-based way using high-fidelity simulation data. Other examples for the

Viktor Grimm
Department of Mathematics and Computer Science, University of Cologne, Weyertal 86-90, 50931 Köln, Germany, e-mail: viktor.grimm@uni-koeln.de

Alexander Heinlein
Delft University of Technology, Delft Institute of Applied Mathematics, Mekelweg 4, 2628 CD Delft, Netherlands e-mail: a.heinlein@tudelft.nl

Axel Klawonn
Department of Mathematics and Computer Science, University of Cologne, Weyertal 86-90, 50931 Köln, Germany e-mail: axel.klawonn@uni-koeln.de
Center for Data and Simulation Science, University of Cologne, Germany

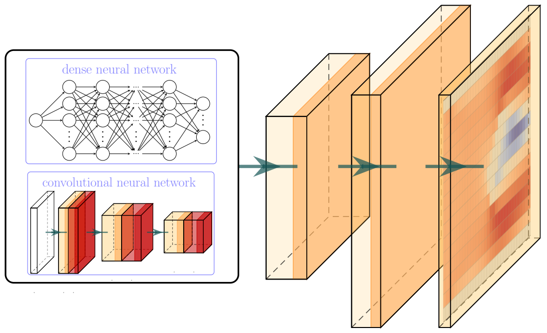


Fig. 1 Exemplary CNN-based surrogate model. The first block transforms the problem parametrization into a low-dimensional representation (latent representation) of the solution, and the right part of the model decodes the corresponding image of the solution field.

CNN approach involve the use of a physics-based loss function, that is, based on the residual of the partial differential equation (PDE), as well as the BCs of the BVP; this is also often denoted as physics-informed or physics-aware machine learning (ML). In particular, in [10], a model for predicting the solutions of the stationary diffusion equation for a single fixed geometry but varying BCs, encoded as an input image, is proposed. In [4], the authors employ a physics-based CNN model for predicting incompressible Navier–Stokes flow in parameterized geometries that is, the exact placement of the boundaries of the geometries depend on a parameter. More recently, the authors of this work have extended the previous approaches to a physics-aware CNN for predicting incompressible Navier–Stokes flow in more general geometries and also varying boundary conditions; cf. [5]. For scientific machine learning (SciML) overview papers with additional references on related approaches, we refer to [1, 13].

In this paper, we will compare the accuracy and convergence of a CNN model, optimized using a (stochastic) gradient descent-type method using a physics-based loss function, with a classical FD discretization, solving the resulting discrete linear system of equations using an (unpreconditioned) conjugate gradient (CG) method, for a simple stationary diffusion problem. In order to focus on these aspects and remove any other complexities, we focus on a single problem configuration, that is, we neglect the encoder part in fig. 1 and focus on training the decoder path. The paper is organized as follows: In section 2, we introduce our stationary diffusion model problem and the simple difference discretization employed. Then, in section 3, we briefly discuss how to solve the resulting discrete system of equations using the CG method as well as how to optimize a CNN model for predicting the same solution. Finally, we compare the performance of both solution frameworks with respect to accuracy and convergence in section 4.

2 Model problem and finite difference discretization

Let us consider a simple stationary diffusion problem on computational domain $\Omega := [0,1]^2$: find a function u, such that

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where f is some right hand side function. We discretize eq. (1) using FDs. In particular, we approximate the Laplacian using the central difference scheme

$$\frac{\partial^2 u}{\partial x^2}(x) \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$$

in each dimension. In particular, we consider a uniform grid $\Omega_h = \left\{(x_i, y_j)_{i,j}\right\}$ with $x_i := ih$ and $y_j := jh$, the step size $h = 1/n$, and $u_{i,j} := u(x_i, y_j)$. This yields a set of linear equations

$$\Delta u(x_i) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}.$$

With $f_{ij} := f(x_i, y_j)$, the discrete form of eq. (1) corresponds to a system of $(n-1)^2$ equations

$$-\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} - \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = f_{i,j} \qquad \forall 0 < i,j < n, \tag{2}$$

where, due to the BCs,

$$u_{i,j} = 0 \qquad \text{for } 0 \leq i,j \leq n \text{ with } i \in \{0,n\} \ \vee \ j \in \{0,n\}.$$

This yields a sparse system of linear equations

$$Au = b. \tag{3}$$

with a symmetric positive definite (SPD) matrix.

3 Solving the model problem using classical methods versus using convolutional neural networks

Efficient classical numerical solvers

Since our model problem, that is, stationary diffusion on the unit square, is arguably one of the most investigated problems for the development of solvers, there is a wide range of efficient solvers for eq. (3). Hence, we keep this discussion rather short. A standard solver for systems with an SPD matrix is the conjugate gradient (CG) method. The convergence of the CG method is determined by the spectrum of the matrix, and in particular, it can be bounded in terms of the condition number of the system matrix A, which scales with $\frac{1}{h^2}$ for our model problem. The h dependence of the convergence of the CG method can be fixed by acceleration using preconditioners, such as domain decomposition [11] and multigrid [12] methods, to name just two popular classes of efficient and scalable preconditioners for eq. (3).

For the purpose of comparing numerical solvers against a closely related ML approach for solving a stationary problem, we will use the CG method without preconditioning as the prototypical solver.

A solver based on convolutional neural networks (CNNs)

Solving eq. (3) corresponds to finding the coefficients $u_{i,j}$, which are structured based on the uniform grid $\Omega_h = \left\{(x_i, y_j)_{i,j}\right\}$. We can simply interpret the discrete solution as a pixel image, with each pixel corresponding to one coefficient in the solution vector u. Hence, in several works, CNNs, which are very effective in image processing, have been trained to learn the discrete solution of a partial differential equation; cf. fig. 1 for a sketch of this approach and the discussion below. In practice, as we will also see in section 4, this approach is not competitive for solving a single BVP. However, when used as a reduced order model for a parametrized model problem (e.g., with respect to the geometry), the higher computing costs for the training can be justified if the solutions of multiple BVPs can be predicted using a single model.

Here, we focus on training a neural network using a physics-informed, sometimes also referred to as physics-aware or physics-constraint, approach. Then, a neural network $\mathcal{NN}$ is trained to minimize the norm of the residual of the differential equation, i. e.,

$$\|\Delta\mathcal{NN} + f\|_{\Omega}^2 + \|\mathcal{NN}\|_{\partial\Omega}^2 \to \min,$$

where $\|\cdot\|_{\Omega}$ and $\|\cdot\|_{\partial\Omega}$ are some norms defined based on collocation points inside the domain Ω and on the boundary $\partial\Omega$. If the output of the neural network corresponds to an image, that is, if the output data is a discrete vector on the uniform grid $\Omega_h = \left\{(x_i, y_j)_{i,j}\right\}$, we can employ an FD scheme to formulate the residual of the PDE, resulting in

$$\|b - A \cdot \mathcal{NN}\|_2^2 \to \min, \tag{4}$$

where the term corresponding to the boundary conditions vanishes since they are hard-coded within the matrix A. Note that this can be efficiently implemented in state-of-the-art ML libraries, such as Tensorflow: the matrix A does not have to be assembled, but it can be applied in a matrix-free fashion by using the FD stencil eq. (2) as a fixed kernel in a convolutional layer and applying it to the output of the network.

We note that solving eq. (4) directly for u is equivalent to solving the least-squares problem corresponding to eq. (3), which amounts to solving the normal equations

$$A^\top A u = A^\top b. \tag{5}$$

The system matrix $A^\top A$ is still SPD, so eq. (5) can also be solved using the CG method. However, the convergence will be much slower, as the condition number

$$\kappa\left(A^\top A\right) = \kappa\left(A\right)^2.$$

The situation is changed further once u is replaced by a neural network $\mathcal{NN}$. Hence, minimizing the loss function with respect to the network parameters θ does not correspond to solving a linear system anymore. Moreover, the loss function is, in general, not even a convex function with respect to the network parameters anymore. Thus, in addition to solving a problem eq. (4) that has a significantly worse conditioning than the original problem eq. (3), we cannot use the CG method let alone another Krylov subspace method anymore.

Minimizing eq. (4) with respect to the network parameters, which is also denoted as training the neural network, is usually performed using either a variant of stochastic gradient descent (SGD), such as the Adam (adaptive moments) optimizer [7], or a second order quasi-Newton method, such as L-BFGS [9]. Those optimizers and their parameters are typically chosen based on heuristics, which clearly shows that, at this point, we have lost most of the properties of the original problem eq. (3) beneficial for a numerical solver.

4 Numerical results

In this section, we compare different solution methods for an FD discretization of eq. (1). In particular, we employ the gradient descent (GD) and conjugate gradient (CG) methods for the original equations eq. (3) as well as the normal equations eq. (5) arising from a least-squares formulation of the problem. We compare those results against training a CNN to predict the coefficient vector using the GD and Adam [7] methods for the physics-informed loss function, which corresponds to the least-squares formulation eq. (4).

It should also be noted that the training performance and prediction accuracy of neural networks strongly depend on the choice of the hyperparameters, which include the specific network architecture and parameters of the optimizer. In advance of our numerical study, we have carried out a detailed hyperparameter optimization to obtain good performance of the CNN models. We have exclude the hyperparameter optimization step from the discussion and only discuss the performance for the optimized hyperparameters; in fact, it is not obvious how to take the hyperparameter optimization into account in the comparison in a fair way.
The issue to be investigated in this work, the efficiency of CNNs for solving PDEs, has arisen in the context of using CNNs as surrogate models, hence the chosen network

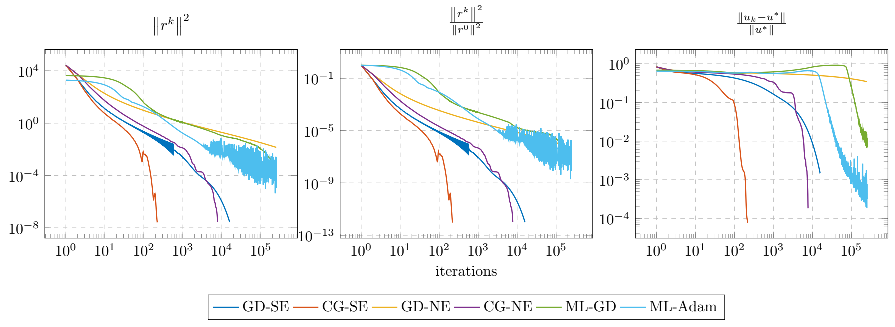


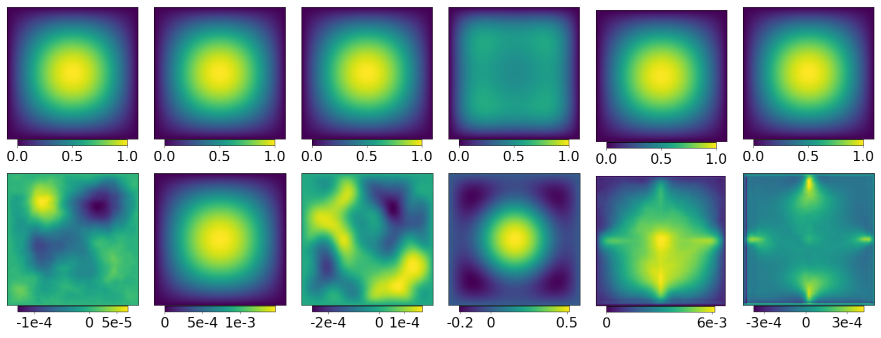
Fig. 2: Convergence of the GD, CG and Adam methods for the original linear equation system eq. (3) and the least-squares problem eq. (4) for the FD discretization u and the CNN u_{NN}. Comparison of the absolute and relative residuum $\|r_k\|^2 \,/\, \|r_0\|^2$ where $r_k = b - Au_k$, and the relative error $\|u_k - u^*\| \,/\, \|u^*\|$.

architecture and hyperparameters are based on this use case; for more details cf. [3, 5]. Specifically, experiments have identified that we obtain optimal results here for GD with a learning rate of 10^{-5} and for Adam with a learning rate of $5.0 \cdot 10^{-5}$. We always use swish as the activation function.

For our experiments, we choose $f = 2\pi^2 \sin(\pi x) \sin(\pi y)$ as the right hand side. The resulting BVP has the analytical solution $u^* = \sin(\pi x) \sin(\pi y)$, which we use as the reference. In this work, we exlusively consider an FD discrezation of the computational domain Ω with $N = 128$ grid nodes in each direction; this results in a total problem size of 16 384 nodes or degrees of freedom, respectively. For the classical methods, we use a fixed but random initial guess, the parameters of the CNNs are randomly initialized using the He normal initialization. We compare the convergence of the methods via the squared relative residual $\|r_k\|^2/\ \|r_0\|^2$, which corresponds to a relative mean squared error (MSE). For the classical numerical methods, we stop the iteration once a tolerance of 10^{-12} for the relative residual or an iteration count of 250 k iterations is reached. The CNNs are always trained for 250 k iterations or epochs.

We compare the relative residuals for the various methods applied to the standard and normal equations in fig. 2. As expected, the CG method applied to the standard equation (CG-SE) converges the fastest after 221 iterations; note again that the convergence could be significantly improved using preconditioning techniques. The CG method applied to the normals equation (CG-NE) converges within 7 737 iterations, the GD method on the original equation (GD-SE) in 15 811 iterations. The GD method on the normal equations (GD-NE) does not converge within 250 k iterations and reaches a relative residual of $5.2 \cdot 10^{-7}$ at termination of the iteration.

As can be seen in fig. 3d, the GD-NE solution, which has not converged within 250 k iterations, has a large relative L_2-error of 34 % compared with the analytical solution. For CG-SE, CG-NE, and GD-SE, we obtain errors of 0.008 %, 0.02 %, and



(a) CG-SE (b) GD-SE (c) CG-NE (d) GD-NE (e) ML-GD (f) ML-Adam

Fig. 3: The solutions (top row) achieved with the various methods and the corresponding erorrs ($u^* - u$) (bottom row) w.r.t the analytical solution at the grid nodes.

0.15 %, respectively, at convergence. In terms of convergence with respect to the relative squared residual norm, the ML approaches perform worse. Both ML-GD and ML-Adam do not achieve a relative tolerance of 10^{-12} and the training is stopped after 250 k iterations/epochs with a final relative residual of $1.5 \cdot 10^{-7}$ for ML-GD and $3.1 \cdot 10^{-8}$ for ML-Adam. Nonetheless, we achieve relative L_2-errors of 0.7 % for ML-GD and 0.02 % for ML-Adam. These are significantly lower than for GD-NE, even though the methods terminate at a similar relative residual. In fact, the accuracy is within one order of magnitude of the CG solutions and even better than the GD-SE solution; cf. also fig. 3.

Let us discuss why, in comparison, the error may be much lower for the CNN compared to the classical numerical solvers for a residual in the same order of magnitude. In particular, for the error e and the residual r, we have

$$Ae = A(u^* - u) = b - Au = r$$

Hence, of course, the relation of $\|e\|$ and $\|r\|$ depends on how the error decomposes into eigenfunctions of high/low eigenvalues. Since the CNNs were able to achieve comparatively low error while exhibiting higher absolute and relative residual, especially compared to the CG solutions, this suggests that the corresponding error is mainly composed of eigenfunctions corresponding to high eigenvalues. In particular, this implies that the CNNs exhibit some form of spectral bias, i.e., that they tend to learn eigenfunctions corresponding to low eigenvalues.

5 Conclusion

In this work, we have compared physics-informed CNNs with classical methods for solvinge PDEs on the example of the stationary diffusion problem. We have shown that solution methods that take advantage of properties of the problem, such as the CG method, outperform the ML approach both in the accuracy achieved and in the speed of convergence. Yet, the ML solutions learned were within an order of magnitude of the CG solutions, i.e., they were not infeasible. But the much slower convergence coupled with the need for hyperparameter optimization as well as the heuristic nature of the choice of method parameters argue for the use of classical methods. Nonetheless, with an ML approach it is possible to include parameters, such as boundary conditions, geometry, etc., as input. In such cases, ML approaches are superior to classical methods and thus there is a sound reason again to use them.

Acknowledgements This work was performed as part of the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE) and received funding from the Helmholtz Association of German Research Centers. We gratefully acknowledge the use of the computational facilities of the Center for Data and Simulation Science (CDS) at the University of Cologne.

References

1. Cuomo, S., di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F.: Scientific machine learning through physics-informed neural networks: Where we are and what's next. arXiv (2022). URL arXiv:2201.05624
2. Eichinger, M., Heinlein, A., Klawonn, A.: Stationary flow predictions using convolutional neural networks. In: Numerical Mathematics and Advanced Applications ENUMATH 2019, pp. 541–549. Springer (2021)
3. Eichinger, M., Heinlein, A., Klawonn, A.: Surrogate convolutional neural network models for steady computational fluid dynamics simulations. Electronic Transactions on Numerical Analysis **56**, 235–255 (2022)
4. Gao, H., Sun, L., Wang, J.: Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. Journal of Computational Physics **428**, 110079 (2021). DOI 10.1016/j.jcp.2020.110079
5. Grimm, V., Heinlein, A., Klawonn, A.: Physics-aware convolutional neural networks for two-dimensional flow predictions. In preparation
6. Guo, X., Li, W., Iorio, F.: Convolutional neural networks for steady flow approximation. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, p. 481–490. Association for Computing Machinery, New York, NY, USA (2016). DOI 10.1145/2939672.2939738
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
8. Lassila, T., Manzoni, A., Quarteroni, A., Rozza, G.: Model order reduction in fluid dynamics: challenges and perspectives. Reduced Order Methods for modeling and computational reduction pp. 235–273 (2014)
9. Liu, D., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical Programming **45**, 503–528 (1989). DOI https://doi.org/10.1007/BF01589116
10. Sharma, R., Farimani, A.B., Gomes, J., Eastman, P., Pande, V.: Weakly-supervised learning of heat transport via physics informed loss. arXiv (2018). URL arXiv:1807.11374
11. Toselli, A., Widlund, O.: Domain decomposition methods-algorithms and theory, vol. 34. Springer Science & Business Media (2004)
12. Trottenberg, U., Oosterlee, C.W., Schuller, A.: Multigrid. Elsevier (2000)
13. Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V.: Integrating scientific knowledge with machine learning for engineering and environmental systems. arXiv (2021). URL arXiv:2003.04919