# Generative Models for Solving Nonlinear Partial Differential Equations

**Ameya Joshi**[*†] **Viraj Shah**[*†] **Sambuddha Ghosal**[‡] **Balaji Pokuri**[‡]
Iowa State University
{viraj, ameya, sghosal, balajip}@iastate.edu

**Soumik Sarkar** [‡] **Baskar Ganapathysubramanian** [‡] **Chinmay Hegde** [†]
Iowa State University, Ames
{soumiks, baskarg, chinmay}@iastate.edu

## Abstract

Partial differential equations (PDEs) describe a wide variety of physical systems. While there exist several numerical methods to solve PDEs, they are often computationally expensive, and solutions to varying boundary conditions and forcing functions need to be derived from scratch. We present a conditional generative modeling based approach to solve families of PDEs parameterized by a distribution of boundary conditions and coefficients. We validate our approach by solving a family of nonlinear PDEs: the Burgers' equation with a single trained model. We also compare with other neural network based solvers as well as standard numerical solvers and demonstrate comparable accuracy while being computationally more efficient.

## 1 Introduction

**Motivation.** Complex physical systems are often characterized using partial differential equations (PDEs). While analytically solving such nonlinear PDEs is generally difficult, there has been great progress in numerical approaches such as finite-element (FEM), finite-volume (FVM), and finite difference (FDM) methods [21]. However, for the case of non-linear PDEs, such algorithms require careful design and domain specific understanding to solve the problem accurately and tractably. In addition, the numerical approaches are often computationally expensive.

We present an efficient alternative approach wherein we train a conditional generative model, that we call *DiffNet*, to solve a family of PDEs parameterized by coefficients and boundary conditions. While there are several recent works in this domain [15, 25, 14, 3], these approaches often require data in the form of samples of the solution space. Additionally, these approaches generally provide pointwise solutions to a specific PDE instance. However, our approach is *data-free* and does not require availability of training examples from the solution space.

**Contributions.** As our primary application, we design DiffNet models to generate solutions for partial differential equations parameterized by boundary conditions and coefficients. We show that DiffNet provides flexible user control over both boundary conditions and PDE coefficients. As an example, we solve the classical non-linear time varying Burgers' Equation [2] in both viscid and inviscid cases and demonstrate that DiffNet provides very competitive results compared to numerical

[*]Equal contribution
[†]Dept. of Electrical and Computer Engineering
[‡]Dept. of Mechanical Engineering

Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada.

solvers. This marks a significant improvement over recently proposed unsupervised learning methods for solving PDEs, such as [25].

**Related Work.** Data driven approaches for solving PDEs such as [1, 23, 17, 4, 15, 16, 12, 13] use simulated solutions to learn surrogate solvers. Sirignano et al. [19] rely on analytically calculated gradients to train a surrogate solver for high dimensional PDEs. Han et al. [8] approximate gradients of the solution using ideas from reinforcement learning. Raissi et al. [15] demonstrate the use of a fully connected neural network to generate pointwise solutions to a non-linear PDE using a data sampled from the solution. Pang et al. [14] and Yang et al. [22] extend this approach to solve fractional and stochastic PDEs respectively.

Farimani et al. [5] propose a standard conditional GAN to generate solutions to the standard transport equation for specific initial conditions. Zhu et al. [24, 25] employ the use a convolutional encode-decoder architectures along with a conditional FLOW [18] model to surrogate a PDE with stochastic coefficients. Similar to our approach, they rely on the use a physics informed loss to train their model. However, our approach uses adversarial generative models instead of normalizing flows and additionally is flexible enough to allow for variation in both initial conditions as well as the coefficients. Hsieh et al. [9] learn neural network based solvers for linear PDEs and show strong convergence guarantees. Greenfield et al. [7] and Katrutsa et al. [10] learn multigrid solvers for a class of PDEs. Li et al. [11] propose a variational network to solve elliptic linear and nonlinear PDEs.

We note that the above listed approaches typically rely on availability of data to learn the PDE solutions. Conversely, our architecture is *data-free* in the sense that it does not require solution instances. Unlike [15], we generate solution fields over the entire domain (geometry) of the PDE rather than pointwise outputs. For quantitative comparison, we consider the deterministic surrogate introduced in [25] for solving Burgers' Equation, where the input to the learning framework are example input fields that obey certain boundary conditions.

## 2 Proposed Model: DiffNet

Following the notation in Hsieh et al. [9], we consider a nonlinear PDE defined as

$$\mathcal{A}_\nu(\mathbf{u}) = f, \;\; \mathcal{B}(\mathbf{u}) = \mathbf{b} \tag{1}$$

where $\mathbf{u}(s)$ is the solution to the PDE over the domain $\Omega \in \mathbb{R}^s$, $\mathcal{A}_\nu$ is the non-linear functional form of the PDE defined by its coefficients $\nu$, and $f$ is a forcing function. Here, $\mathcal{B}(\cdot)$ refers to the boundary conditions for the PDE.

For solving the above PDE numerically, the standard approach is to discretize $\Omega$ to (say) a square grid, $\mathbf{S} \in \mathbb{D}^s$ where $\mathbb{D}$ is a discrete subspace of $\mathbb{R}^s$. Subsequently, $\mathbf{u}$ can be discretized into a vector, $\bar{\mathbf{u}}$, by approximating via a basis of piecewise-constant functions over each sufficiently small discrete element. Similarly, the boundary conditions are also discretized over the grid. Following the standard approach defined by FEM or FDM methods, we can locally approximate $\mathcal{A}$ using a linear operator $\mathbf{A}$ over each discrete element. The problem then simplifies to iteratively solving a large set of (local) linear equations of the form:

$$\mathbf{A}(\bar{\mathbf{u}}) = 0,$$

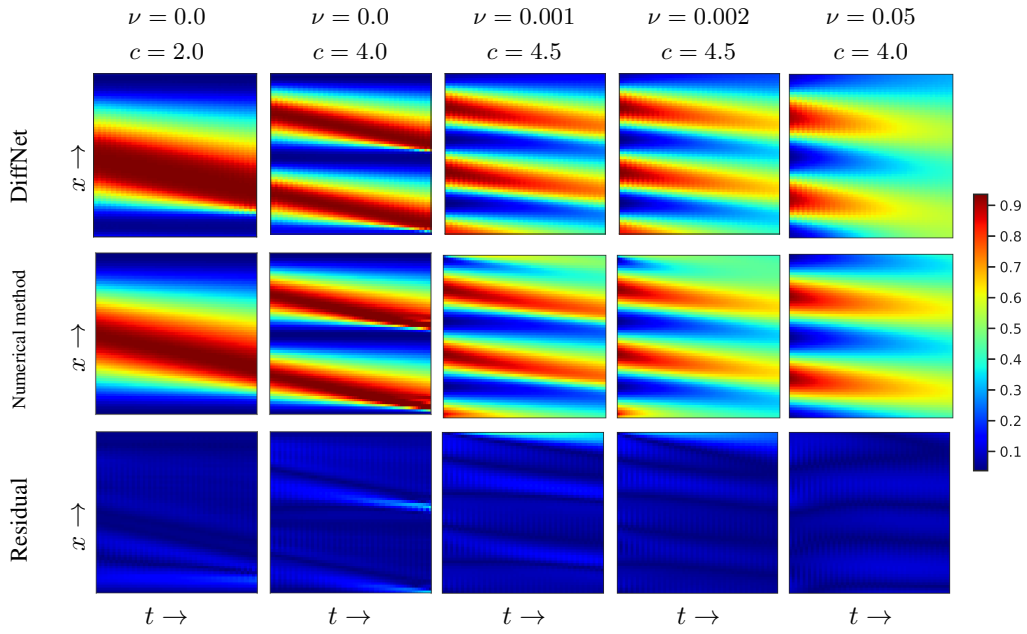
in order to estimate $\mathbf{u}$. Solving each linear system incurs high computational expense but on the other hand, the forward operator $\mathbf{A}(\bar{\mathbf{u}})$ is easier to apply.

In order to model the solution space, we propose a generative neural network that we call *DiffNet*. DiffNet consists of a generator $G_\theta : \mathbb{R}^k \rightarrow \mathbb{R}^d$ that takes as input the boundary conditions $\mathbf{b}$ and any PDE coefficients $\nu$. The generator is then trained to generate the solution to the PDE that corresponds to the input boundary conditions and coefficients. This also models the stochastic case where $\mathbf{b}$ and $\nu$ are sampled from distributions themselves.

We observe that for $G_\theta(\cdot)$ to successfully represent the solution space of the PDE, generator outputs must satisfy two conditions: (1) $G_\theta(\cdot)$ must satisfy the PDE, and (2) $G_\theta(.)$ must respect the provided input boundary conditions. The training loss can therefore be written in terms of two components:

$$L_p(\theta) = \mathbb{E}_{\mathbf{b},\nu}[\|\mathcal{A}_\nu(G_\theta(\mathbf{b},\nu)) - f\|_2^2], \quad L_b(\theta) = \mathbb{E}_{\mathbf{b}}\|\mathcal{B}(G_\theta(\mathbf{b},\nu)) - \mathbf{b}\|_2^2] \tag{2}$$

The first term, $L_p$, minimizes the residual of the PDE while the second term, $L_b$, pushes the generator to learn to reproduce the given boundary conditions. In order to train the above network, we sample a




| | | | | | |
|---|---|---|---|---|---|
| Relative $\ell_2$ error | 0.073 | 0.095 | 0.147 | 0.117 | 0.071 |

Figure 1: *Example of solutions to Burger's Equation generated using DiffNet for a family of boundary conditions. Row (a) shows solutions generated by DiffNet. Row (b) are actual solutions calculated by explicit Euler methods. Row (c) represents the residual between the two. Column (1) explicitly shows that our model generalizes even for an unseen initial condition,* $c = 2.0$.

batch of boundary conditions and coefficients, $\{\mathbf{b}_i, \nu_i\}, i = \{1, 2, \cdots, k\}$ and alternately optimize each loss term with respect to $\theta$ using stochastic gradient descent (or a variant such as Adam). Using minibatches sampled from a distribution of $\mathbf{b}$ and $\nu$ allows the generator to learn the solutions for the family of PDEs parameterized over $(\mathbf{b}, \nu)$.

**Implementing the forward model.** The derivatives of $L_p(\theta)$ with respect to $\theta$ require calculating $\frac{\partial \mathcal{A}_\nu}{\partial \theta}$. This is generally non-trivial and to make this tractable we borrow ideas from finite difference methods. We approximate the $k^{th}$ order derivative operator, $\nabla^k_{(\mathbf{x},t)}$ with convolutional operators defined using finite-difference kernels. In practice, we use $3 \times 3$ Sobel kernels [20] for first order derivatives and Laplacian kernels [6] for second order derivatives. This is identical to the approach adopted by Zhu et al. [25]; however, their setup is somewhat restrictive since they use Encoder-Decoder (ED) networks to construct solutions for a given specific PDE.

In the casesof time-dependent PDEs, the generator $G_\theta$ must learn to first reproduce the initial condition $\mathbf{u}_0$ at $t = 0$ in order to successfully generate the rest of the solution. This informs our alternating minimization approach: we alternately minimize the boundary loss $L_b$ and the PDE loss $L_p$. While this is not strictly necessary as we can also minimize the Lagrangian form, we note that, in practice, the convergence of the network is highly sensitive to the choice of the Lagrangian coefficient. An incorrect choice leads to failure either by the model learning to generate the trivial solution, 0 or failing to converge. However, in our experience using the alternating approach always leads to successful training.

The advantage of training a conditional generative model such as DiffNet is that we only need to train a *single* model for a distribution of parameters characterizing the system. Our approach allows for interpolating and (possibly) extrapolating over unseen boundary conditions and coefficients to generate solutions.

**Experiments.** We demonstrate that the solution set of partial differential equations (PDEs) describing dynamics of physical systems can be accurately recovered using a DiffNet model. We mention

that data-driven deep generative models have already been proposed as fast surrogates to traditional PDE solvers [25, 24]. However, DiffNet provides an alternate, data-free approach to solving PDEs.

We demonstrate this via a simple non-linear PDE called Burgers' Equation [2]. We use a conditional input $\mathbf{c}$ to DiffNet to control the solution set of Burgers'. Unlike [25], a *single* well-trained DiffNet can generate solutions corresponding to a variety of boundary conditions as well as varying physical parameters.

We first consider the inviscid form of Burgers' Equation. This is a non-linear PDE encountered in fluid mechanics and nonlinear acoustics. The equation, defined in Eq. 3, assumes a non-diffusive fluid through which a wave with initial state $f_i$ is passed. Let $\mathbf{U}(x,t) = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_n]$ be a particular solution of the Burger's equation. Then,

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \odot \frac{\partial \mathbf{U}}{\partial x} = 0, \quad \mathbf{U}(x,0) = f_i(x). \quad (3)$$

Suppose we model the field $\mathbf{U}(x,t)$ as an image (where rows correspond to space and columns correspond to time; see Fig 1). We train a variant of DiffNet that generates solutions $G_\theta(\mathbf{z})$ to Eq. 3 for a given boundary condition as input.

In our experiments, we set $\mathbf{b}$ as a (discretized) raised-cosine function parameterized on $c$. We sample $\mathbf{b}$ uniformly from set $B$ given as:

$$B = \{x | x = \frac{1}{2}\left(1 - \cos(2\pi \mathbf{x} c/d)\right), c \in [3,6]\}\}.$$

Table 1: *Relative $\ell_2$ error of solutions generated by DiffNet compared to that of ED networks. The error is calculated with respect to numerical solutions. Note that a single trained DiffNet shows comparable performance to a separate ED networks[25].*

| Viscosity coefficient $\nu$ | Frequency of boundary conditions ($c$) | DiffNet | ED [25] |
|---|---|---|---|
| 0.002 | 2.0 | 0.08 | 0.04 |
| | 4.0 | 0.13 | 0.08 |
| | 5.5 | 0.29 | 0.30 |
| | 6.0 | 0.20 | 0.15 |
| 0.006 | 2.0 | 0.08 | 0.04 |
| | 4.0 | 0.10 | 0.14 |
| | 5.5 | 0.25 | 0.29 |
| | 6.0 | 0.15 | 0.21 |
| 0.02 | 2.0 | 0.08 | 0.08 |
| | 4.0 | 0.07 | 0.14 |
| | 5.5 | 0.18 | 0.23 |
| | 6.0 | 0.09 | 0.16 |
| 0.03 | 2.0 | 0.08 | 0.09 |
| | 4.0 | 0.06 | 0.12 |
| | 5.5 | 0.17 | 0.20 |
| | 6.0 | 0.08 | 0.12 |

We show in Fig. 1 that DiffNet successfully learns to generate solutions for the family of boundary conditions given by $B$.

**Viscid Burgers' equation.** We extend the above algorithm to generate solutions for a *family* of PDEs, with a physical scalar parameter indexing each PDE. We consider the family of viscid Burger's equation represented in Eq. 4 with the viscosity term, $\nu$:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \odot \frac{\partial \mathbf{U}}{\partial x} = \nu \frac{\partial^2 \mathbf{U}}{\partial x^2}. \quad (4)$$

Similar to the inviscid case, we provide both the boundary condition $\mathbf{b}$ and the coefficient of viscosity $\nu$ as input to the generator. Our input vector becomes $\mathbf{z} = \mathbf{c} = [\mathbf{b}, \nu]$, where $\mathbf{b}$ is sampled uniformly from $B$, and $\nu$ is sampled randomly from the set Unif$[0.001, 0.05]$. Fig. 1 depicts the solutions corresponding to different combinations of $\mathbf{b}$ and $\nu$.

We compare our DiffNet-based surrogate solutions to numerical solutions (computed using explicit Euler methods) for both the viscid and inviscid cases in Fig. 1, and observe that our model is accurate. Moreover, DiffNet shows effective generalization by generating solutions for boundary conditions that were *not* used during training. While the results are preliminary, DiffNet shows promise towards generalizing for a variety of boundary conditions and coefficients.

## 3 Discussion and Conclusion

We introduce a generative model, DiffNet that learns to solve families of PDEs for a variety of boundary conditions. We show comparable results to both numerical PDE solvers as well as other approach deep network based surrogate solvers.

Several potential directions of research remain. Given that generative models, such as GANs, can model arbitrary probability distributions, our models could be further extended to solve stochastic PDEs, which are more complex. Additionally, limiting the generator to images restricts its use to problems in two dimensions. A compelling extension would be to extend InvNets for other domains such as graphs.

## References


[1] Cosmin Anitescu, Elena Atroshchenko, Naif Alajlan, and Timon Rabczuk. Artificial neural network methods for the solution of second order boundary value problems. *Computers, Materials & Continua*, 59(1):345–359, 2019.

[2] Harry Bateman. Some recent researches on the motion of fluids. *Monthly Weather Review*, 1915.

[3] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

[4] Mulin Cheng, Thomas Y Hou, Mike Yan, and Zhiwen Zhang. A data-driven stochastic method for elliptic pdes with random coefficients. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):452–493, 2013.

[5] Amir Barati Farimani, Joseph Gomes, and Vijay S. Pande. Deep learning the physics of transport phenomena. *arxiv preprint arXiv:1709.02432*, 2017.

[6] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006.

[7] Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid pde solvers. In *ICML*, 2019.

[8] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[9] Jun-Ting Hsieh, Shengjia Zhao, Stephan Eismann, Lucia Mirabella, and Stefano Ermon. Learning neural PDE solvers with convergence guarantees. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2019.

[10] Alexandr Katrutsa, Talgat Daulbaev, and Ivan Oseledets. Deep multigrid: learning prolongation and restriction matrices. *arXiv preprint arXiv:1711.03825*, 2017.

[11] Yingzhou Li, Jianfeng Lu, and Anqi Mao. Variational training of neural network approximations of solution maps for physical models. *arXiv preprint arXiv:1905.02789*, 2019.

[12] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *ICLR*, 2017.

[13] Craig Michoski, Milos Milosavljevic, Todd Oliver, and David Hatch. Solving irregular and data-enriched differential equations using deep neural networks. *arXiv preprint arXiv:1905.04351*, 2019.

[14] Guofei Pang, Lu Lu, and George Em Karniadakis. fPINNs: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[15] M. Raissi, P. Perdikaris, and G.E. Karniadakis. 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.

[16] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[17] F Regazzoni, L Dedè, and A Quarteroni. Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics*, 397:108852, 2019.

[18] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proc. Int. Conf. Machine Learning (ICML)*, 2016.

[19] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[20] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing, 1968.

[21] Eitan Tadmor. A review of numerical methods for nonlinear partial differential equations. *Bulletin of the American Mathematical Society*, 49(4):507–554, 2012.

[22] Liu Yang, Dongkun Zhang, and George Em Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *arXiv preprint arXiv:1811.02033*, 2018.

[23] Kyongmin Yeo. Data-driven reconstruction of nonlinear dynamics from sparse observation. *Journal of Computational Physics*, 2019.

[24] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

[25] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *arXiv preprint arXiv:1901.06314*, 2019.