LEARNING TO SOLVE DIFFERENTIAL EQUATIONS ACROSS INITIAL CONDITIONS

Shehryar Malik[*]
Information Technology University
Lahore, Pakistan
`shehryar.malik@itu.edu.pk`

Usman Anwar[*]
Information Technology University
Lahore, Pakistan
`usman.anwar@itu.edu.pk`

Ali Ahmed
Department of Electrical Engineering
Information Technology University
Lahore, Pakistan
`ali.ahmed@itu.edu.pk`

Alireza Aghasi
J. Mack Robinson College of Business
Georgia State University
Atlanta, GA, USA.
`aaghasi@gsu.edu`

ABSTRACT

Recently, there has been a lot of interest in using neural networks for solving partial differential equations. A number of neural network-based partial differential equation solvers have been formulated which provide performances equivalent, and in some cases even superior, to classical solvers. However, these neural solvers, in general, need to be retrained each time the initial conditions or the domain of the partial differential equation changes. In this work, we posit the problem of approximating the solution of a fixed partial differential equation for any arbitrary initial conditions as learning a conditional probability distribution. We demonstrate the utility of our method on Burger's Equation.

1 INTRODUCTION

Partial differential equations (PDEs) are of great importance in various fields such as science, engineering and economics. However, despite this, it is generally not possible to obtain analytic solutions for them. Instead, one has to resort to numerical schemes for approximating these solutions. However, these numerical schemes are both slow and computationally intensive especially for higher dimensions. Furthermore, the higher the dimension is, the greater is the error in the calculation of the derivatives required to approximate the solution.

Recently, Raissi et al. (2019) proposed to use neural networks to approximate solutions of PDEs. They do so by forcing neural networks to produce outputs that satisfy the PDE. The derivatives required to enforce this condition are computed using automatic differentiation and are exact up to the precision of the computing machine. Furthermore, this approach is *mesh free* i.e. one does not need to discretize the domain of the solution as required in methods such as finite element analysis.

In parallel, we have seen tremendous improvements in the generalization capacities of machine learning methods such as generative adversarial networks and flow-based models. These models have demonstrated a remarkable ability to capture probability distributions even when only representative samples are available [Kingma & Dhariwal (2018), Brock et al. (2019), Oord et al. (2016)].

In this work, we generalize the framework of Yang & Perdikaris (2018) over the distribution of initial conditions and demonstrate that even if the model is trained on a subset of them, it is able to generalize well and produce solutions for any arbitrary initial conditions. Furthermore, our model also provides uncertainty quantifications which loosely correlate with the error in the solution.

This paper is structured as follows: in Section 2, we provide an overview of related works on partial differential equations. Section 3 presents our method and Section 4 talks about our experiments and results. Finally, Section 5 concludes the paper.

---

[*]Both authors contributed equally.

2 RELATED WORKS

PDEs have been traditionally solved through numerical methods such as finite differences, finite element methods and spectral methods (see for e.g. Tadmor (2012)). Lagaris et al. (1998) made the first attempt at using neural networks to approximate solutions of PDEs. However, the idea did not catch up. The recent renaissance of neural networks and its remarkable successes in computer vision and natural language processing [Krizhevsky et al. (2012), Devlin et al. (2018)] have sparked a new interest in scientific machine learning i.e. in applying data driven methods to problems in natural sciences [Han et al. (2018), Senior et al. (2020), Iqbal et al. (2019)]. Since a huge number of problems in the natural and physical sciences are described by PDEs, this has inevitably aroused interest in using machine learning to solve PDEs. In this regard, the physics-informed neural network method in Raissi et al. (2019) was a critical development. It made two major contributions: (a) it showed that the automatic differentiation machinery of machine learning frameworks such as TensorFlow allowed the computation of partial derivatives to a higher order of accuracy than finite differences, and (b) it empirically showed that neural networks can be used to approximate solutions of PDEs by simply forcing them to produce outputs that satisfy the PDE. This framework has since been extended to fractional and stochastic PDEs [Pang et al. (2018), Zhang et al. (2019)].

Neural network based solvers do not, in general, provide any convergence guarantees. Hence, it is important to have an uncertainty estimate for the solution produced by these methods. Yang & Perdikaris (2018) and Yang & Perdikaris (2019) formulate the method in Raissi et al. (2019) in the framework of generative adversarial networks [Goodfellow et al. (2014)] and show that it is possible to obtain uncertainty estimates as well.

Concurrent to this has been considerable work on combining various classical solvers with machine learning methods to obtain more efficient solvers such as combining the Runge Kutta method with convolutional networks in Zhu et al. (2018) and the Galarkin method with deep learning in Sirignano & Spiliopoulos (2018).

However, one area which has attracted considerably less attention is attempting to learn a 'general' PDE solver through neural networks. General, in this context, refers to a PDE solver which does not need to be retrained if one or more constraints such as the initial or boundary condition or the domain are changed. Rather, the general PDE solver relies on the generalization capacity of neural networks to approximate the solution of a PDE across all possible sets of constraints. In this regard, Hsieh et al. (2019) and Farimani et al. (2017) have, for example, demonstrated that it is possible to learn a general PDE solver for some simple linear and elliptic PDEs. However, developing a single general PDE solver for all types of PDEs (parabolic, elliptic and hyperbolic) still remains an open problem.

3 METHODOLOGY

We consider partial differential equations (PDEs) of the form

$$\frac{\partial u(\mathbf{x},t;i)}{\partial t} = f\left(\mathbf{x},t,u(\mathbf{x},t;i),\frac{\partial u(\mathbf{x},t;i)}{\partial \mathbf{x}},\ldots,\frac{\partial^d u(\mathbf{x},t;i)}{\partial \mathbf{x}^d}\right) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^p$, $t \in \mathbb{R}$, $u : \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}$ is the solution of the PDE and f is a known function. Since the solution u of this PDE depends on the initial conditions i, we make this dependence explicit by writing $u(\mathbf{x},t;i)$. We denote the distribution of initial conditions with $\mathcal{I}$.

We propose to train a single Generative Adversarial Network (GAN) [Goodfellow et al. (2014)] for solving a PDE of the form of Equation 1 for any initial conditions drawn from the distribution $\mathcal{I}$. Note that the PDE is fixed (i.e. f is fixed) and only the initial conditions are varied. The generator G_θ takes in as input a random noise vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{1})$, $\mathbf{x}$, t and a particular instance of the initial conditions $i \sim \mathcal{I}$ and generates an approximation of $u(\mathbf{x},t;i)$. The discriminator D_ψ takes two tuples $(\mathbf{x}_1,t_1,G_\theta(\mathbf{x}_1,t_1,i_1,\mathbf{z}_1),i_1)$ and $(\mathbf{x}_2,t_2,u(\mathbf{x}_2,t_2;i_2),i_2)$ and is asked to identify the tuple generated by the generator (as in a typical GAN setup).

The generator G_θ is composed of three networks: the encoder, approximator and reconstructor. The encoder takes in as input a particular instance of the initial conditions $i \sim \mathcal{I}$ and encodes it into some latent vector v. The approximator then takes in as input the latent vector v along with

some spatio-temporal coordinates $(\mathbf{x}, t)$ and outputs an approximation of $u(\mathbf{x}, t; i)$. Finally, the reconstructor takes in as input the outputs of the approximator for several different spatio-temporal coordinates but for the same initial conditions and tries to reconstruct the latent representation of the initial conditions v from these approximations. The reconstructor (inspired, in part, by the InfoGAN architecture [Chen et al. (2016)]) forces the approximator to condition *all* of its outputs on the initial conditions, since otherwise the approximator may learn to produce samples from just one field irrespective of the initial conditions provided to it.

In addition to minimizing the typical GAN loss [Goodfellow et al. (2014)] given by

$$\mathcal{L}_{GAN} = \mathbb{E}_{i,\mathbf{x},t,\mathbf{z}} \left[\log D_\psi \left(\mathbf{x}, t, u(\mathbf{x}, t; i), i \right) \right] + \mathbb{E}_{i,\mathbf{x},t,\mathbf{z}} \left[1 - \log \left(D_\psi \left(\mathbf{x}, t, G_\theta(\mathbf{x}, t, i, \mathbf{z}), i \right) \right) \right], \quad (2)$$

we force the samples produced by the generator to satisfy Equation 1 by minimizing the residual error

$$\mathcal{L}_{PDE} = \mathbb{E}_{i,\mathbf{x},t,\mathbf{z}} \left[\left\| \left\| \frac{\partial G_\theta(\cdot)}{\partial t} - f \left(\mathbf{x}, t, G_\theta(\cdot), \frac{\partial G_\theta(\cdot)}{\partial \mathbf{x}}, \ldots, \frac{\partial^d G_\theta(\cdot)}{\partial \mathbf{x}^d} \right) \right\| \right\|_2^2 \right] \quad (3)$$

where $\cdot$ is a placeholder for $(\mathbf{x}, t, i, \mathbf{z})$.

As noted before, the derivatives can readily be obtained via automatic differentiation.

Furthermore, we also force the true and approximated initial conditions to be equal by minimizing

$$\mathcal{L}_{IC} = \mathbb{E}_{i,\mathbf{x},\mathbf{z}} \left[\|u(\mathbf{x}, t = 0; i) - G(\mathbf{x}, t = 0, i, \mathbf{z})\|_2^2 \right]. \quad (4)$$

Note that t is fixed here.

Next, we enforce the boundary conditions. This can vary from equation to equation. For Burger's equation we enforce the following

$$\mathcal{L}_{BC} = \mathbb{E}_{t,i,\mathbf{z}} \left[\|G(\mathbf{x}_{UB}, t, i, \mathbf{z}) - G(\mathbf{x}_{LB}, t, i, \mathbf{z})\|_2^2 \right]$$
$$+ \mathbb{E}_{t,i,\mathbf{z}} \left[\left\| \left\| \frac{\partial G(\mathbf{x}_{UB}, t, i, \mathbf{z})}{\partial \mathbf{x}} - \frac{\partial G(\mathbf{x}_{LB}, t, i, \mathbf{z})}{\partial \mathbf{x}} \right\| \right\|_2^2 \right] \quad (5)$$

where $\mathbf{x}_{UB}$ and $\mathbf{x}_{LB}$ denote the upper and lower boundary values of $\mathbf{x}$ respectively.

To train the reconstuctor R, we minimize

$$\mathcal{L}_r = \mathbb{E}_{i,\mathbf{x},t,\mathbf{z}} \left[\|i - R(G(x, t, i, z))\|_2^2 \right]. \quad (6)$$

Our overall objective function is therefore given by

$$\mathcal{L} = \mathcal{L}_{GAN} + \alpha \mathcal{L}_{PDE} + \beta \mathcal{L}_{IC} + \gamma \mathcal{L}_{BC} + \delta \mathcal{L}_r \quad (7)$$

where α, β and γ are scalars.

4 EXPERIMENTS

In this section, we demonstrate our model on the Burger's equation which is given by

$$\frac{\partial u}{\partial t} = 0.1 \frac{\partial^2 u}{\partial x^2} - u \frac{\partial u}{\partial x} \quad (8)$$

for initial conditions $u(x, t = 0)$ of the form $a \sin(bx + c\pi) + d$ where a, b, c and d are real numbers.

We generate solutions for 120 different initial conditions by choosing different combinations of a, b, c and d from the sets $\{-1, 1\}$, $\{0.1, 0.5, 0.9, 1.3, 1.7\}$, $\{0, 0.5, 0.9, 1.0\}$ and $\{0, 0.7, 1.4\}$ respectively. 20 of these solutions constitute the test set (which is kept fixed for all experiments). The rest of the 100 fields are randomly split for each experiment into training and validation sets in the ratio 85:15. The data is generated via the Chebfun package for MATLAB [Driscoll et al. (2014)] for solving PDEs. We discretize $x \in \{-8, 8\}$ into 512 points and $t \in \{0, 10\}$ such that the total number of points are around $100,000$.
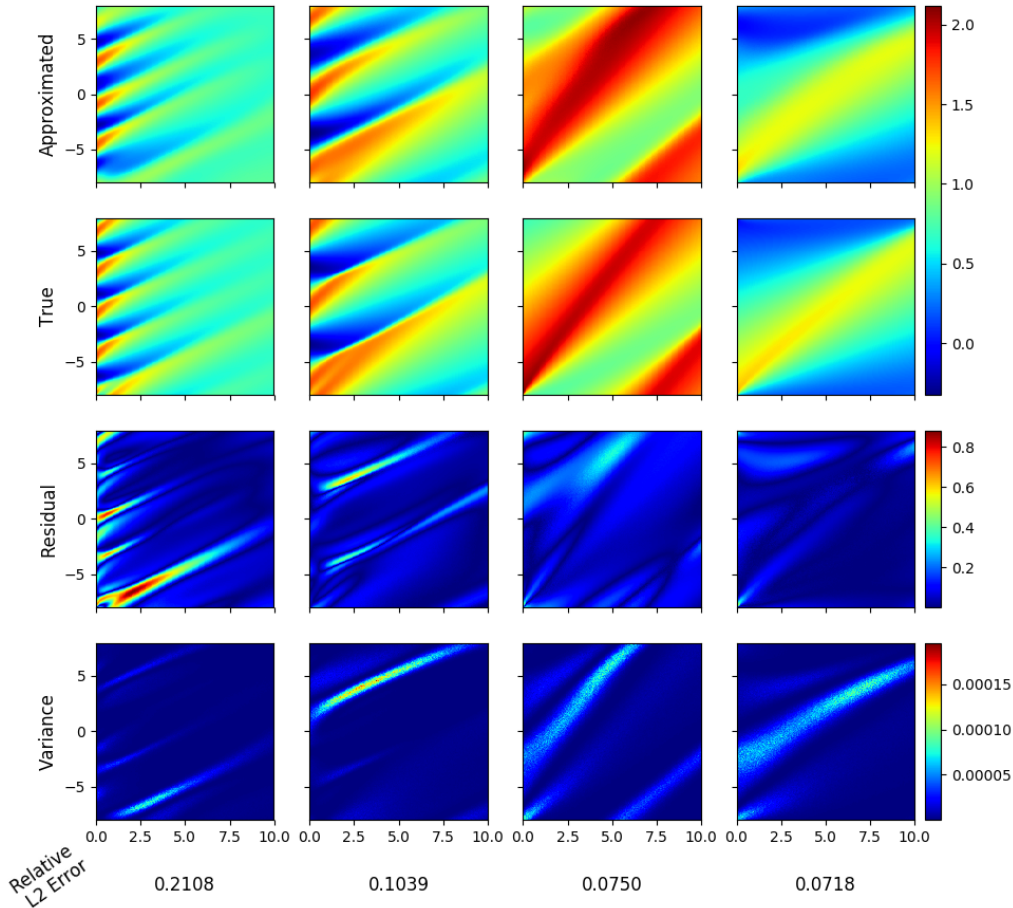
Figure 1: Results on some test fields.

We concatenate $u(x, t = 0)$ at each of the 512 values for x into a vector. These are our initial conditions. We feed these initial conditions into our model as described in the previous section. Table 1 describes the model architecture. The reconstructor is fed the approximator's outputs at 64 different spatio-temporal coordinates. We trained the model parameters using the Adam optimizer with a learning rate of 0.001 for 1.6 million iterations. At each iteration, we feed the model 64 initial points to enforce $\mathcal{L}_{IC}$, 64 boundary points to enforce $\mathcal{L}_{BC}$ and 512 colocation points to enforce $\mathcal{L}_{PDE}$ and $\mathcal{L}_{GAN}$.

We achieved a relative ℓ_2 error of 0.136 on the test set. Figure 3 shows some of the fields generated by the model for initial conditions in the test set. Since the latent variable z in the generator is a random variable, each time the model is run, it generates a slightly different field. As such, we also plot the variance in the model. Note that the variance loosely correlates with the residual (error) field and that both of these are high in regions close to discontinuities. This means that even though our model does not entirely approximate discontinuous regions correctly, it is nevertheless successful in identifying these areas.

While these results are preliminary, they do show the effectiveness of our method as a general PDE solver.

Table 1: Model architecture

	Hidden			
	Layers	Neurons	Activation	Output Neurons
Encoder	2	128	ReLU	32
Approximator	5	256	Tanh	256
Reconstructor	3	256	Tanh	256
Discriminator	3	256	ReLU	256

5 CONCLUSION

We have demonstrated that a generative model can learn to predict solutions for unseen initial conditions when trained only a subset of them for a given PDE. This is the first step towards learning a general neural PDE solver which can provide reliable solutions for novel initial conditions. Furthermore, we have shown that as a result of the probabilistic nature of our model, uncertainty estimates also become freely available and, in our observation, correspond loosely with the error in the approximated solution of the PDE.

REFERENCES

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

T. A Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014. URL http://www.chebfun.org/docs/guide/.

Amir Barati Farimani, Joseph Gomes, and Vijay S Pande. Deep Learning the Physics of Transport Phenomena. 94305, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1718942115.

Jun-Ting Hsieh, Shengjia Zhao, Stephan Eismann, Lucia Mirabella, and Stefano Ermon. Learning neural PDE solvers with convergence guarantees. In *International Conference on Learning Representations*, 2019.

Asim Iqbal, Romesa Khan, and Theofanis Karayannis. Developing a brain atlas through deep learning. *Nature Machine Intelligence*, 1(6):277–287, 2019.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10215–10224. Curran Associates, Inc., 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 2016. URL `http://arxiv.org/abs/1609.03499`.

Guofei Pang, Lu Lu, and George Em Karniadakis. fPINNs: Fractional Physics-Informed Neural Networks. pp. 1–29, 2018. URL `http://arxiv.org/abs/1811.08967`.

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. ISSN 10902716. doi: 10.1016/j.jcp.2018.10.045.

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W R Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020. ISSN 0028-0836. doi: 10.1038/s41586-019-1923-7.

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

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

Yibo Yang and Paris Perdikaris. Physics-informed deep generative models. *arXiv preprint arXiv:1812.03511*, 2018.

Yibo Yang and Paris Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019. ISSN 10902716. doi: 10.1016/j.jcp.2019.05.027.

Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

Mai Zhu, Bo Chang, and Chong Fu. Convolutional neural networks combined with runge-kutta methods. *arXiv preprint arXiv:1802.08831*, 2018.