5 Neural Networks Can Solve PDEs (apparently)

A couple days ago, I was sent a tweet from my friend Chinmay about neural networks being able to solve PDEs (we just finished a PDEs class). Due to me being basically immobile and bedridden, I became aware that I have nothing better to do, I decided to implement one of these. These types of neural networks are called Physics-Informed Neural Networks (PINNs) [1].

Let's define a simple PDE to show how this works:

$$\begin{cases} u_t = 9u_{xx}, & x \in \mathbb{R},\ 0 < t < 1, \\ u(x,0) = e^{-x^2} \end{cases}$$

This is known as the 1D heat equation. To solve this, the idea is to essentially define a neural network with differentiable activation functions $\mathcal{N}_\theta(x,t) \approx u(x,t)$ which approximates the solution. Let's denote $\hat{u}(x,t) \coloneqq \mathcal{N}_\theta(x,t)$. Since the network is differentiable, we can compute $\hat{u}_t = \frac{\partial \mathcal{N}_\theta}{\partial t}$ and $\hat{u}_{xx} = \frac{\partial^2 \mathcal{N}_\theta}{\partial x^2}$ easily. We now need to train the parameters θ such that we minimize the residual, that is, we minimize the error between $\hat{u}_t$ and $9\hat{u}_{xx}$. Mathematically, this is

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_f} \sum_{i=1}^{N_f} |f_\theta(x_i, t_i)|^2$$

where (x_i, t_i) are sampled points in the domain and $f_\theta(x_i,t_i) = \hat{u}_t(x_i,t_i) - 9\hat{u}_{xx}(x_i,t_i)$ is the PDE residual (difference between LHS and RHS of the PDE) at those points.

We also need to learn the initial condition, so we can define another loss function as follows:

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} |\hat{u}(x_i, 0) - e^{-x_i^2}|^2$$

Now, we define our final loss

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{IC}}$$

and all we need to do is to optimize θ such that $\mathcal{L}$ is minimized. Note that if we had boundary conditions, we would add yet another loss, $\mathcal{L}_{\text{BC}}$, to account for that as well.

In practice, to solve this, I defined a fully connected neural network with input (x,t) and output $\hat{u}(x,t)$, given by:

$$\begin{aligned} z^{(0)} &= \begin{bmatrix} x \\ t \end{bmatrix}, \\ z^{(l)} &= \tanh(W^{(l)} z^{(l-1)} + b^{(l)}), \quad \text{for } l = 1, 2, 3, \\ \hat{u}(x,t) &= W^{(4)} z^{(3)} + b^{(4)}, \end{aligned}$$

where each $z^{(l)} \in \mathbb{R}^{64}$ and tanh is applied element-wise.

The network parameters are optimized by minimizing the loss function above using the Adam optimizer. I also used PyTorch's `autograd.grad()` feature to compute the derivatives. Here are my results (at $t = 0.25$), it's crazy how this literally works:

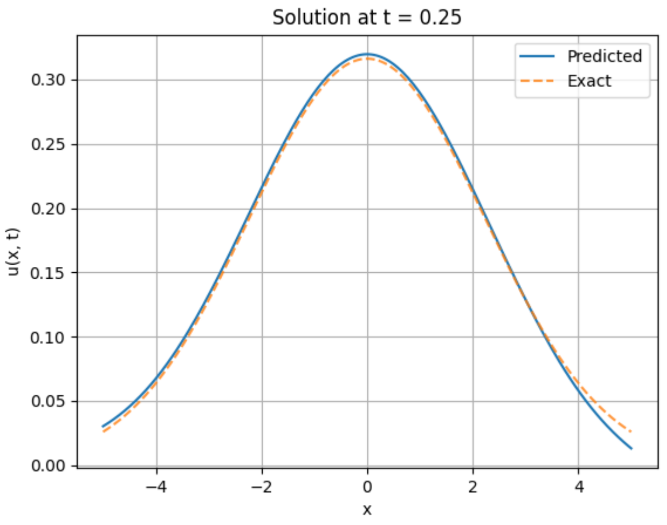

Figure 4: Comparison between the predicted solution $\hat{u}(x,t)$ obtained from the PINN and the exact solution of the 1D heat equation on the infinite domain at time $t = 0.25$. The initial condition is $u(x,0) = e^{-x^2}$ and the exact solution is given by $u(x,t) = \frac{1}{\sqrt{1+36t}} \exp\left(-\frac{x^2}{1+36t}\right)$.

References

[1] 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". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.