G Nikhil Chand

Email- 21f1003825@ds.study.iitm.ac.in

Physics-Informed Neural Networks (PINNs) for Fluid Dynamics

September 19, 2024

Overview

This project leverages Physics-Informed Neural Networks (PINNs) to model fluid dynamics using synthetic data. The neural network is trained to solve the Navier-Stokes equations governing fluid flow, ensuring that the model respects both observed data and physical laws. The primary goal is to predict the velocity field (u,v) and pressure field (p) of a fluid over time using machine learning techniques. The model was implemented in TensorFlow/Keras, with the loss function incorporating the continuity and momentum equations from fluid dynamics

Technology Stack:

- **Programming Language:** Python
- **Frameworks:** TensorFlow, Keras
- **Libraries:** NumPy, Matplotlib
- **Tools:** google Colab
- **Key Machine Learning Concepts:** Physics-Informed Neural Networks (PINNs), Deep Learning, Gradient-Based Optimization (Adam)

Challenges and Solutions:

1. **Incorporating Physics-Based Constraints::** Implementing the physical laws (Navier-Stokes equations) into the neural network required designing custom loss functions that balance data accuracy and physics adherence.

Solution: A custom loss function was developed that computes the partial derivatives of the predicted fields using TensorFlow's GradientTape, ensuring that the model respects the continuity and momentum equations.

2. **Gradient Calculation for Complex Loss Function:** Calculating gradients with respect to multiple input variables (space and time) was computationally expensive and prone to errors in early attempts.

Solution: Used TensorFlow's automatic differentiation to track gradients for velocity and pressure fields, which simplified the implementation

Team Size:

This was an **individual** project, where I was responsible for all aspects

Project Outcome:

The PINN model successfully learned to predict the velocity and pressure fields that adhere to the Navier-Stokes equations. The model was trained on synthetic data and achieved low data and physics-informed loss, demonstrating the effectiveness of PINNs in solving fluid dynamics problems. Key results include:

1. Accurate predictions of the velocity field (u,vu, vu,v) and pressure field (ppp) over time.
2. Visualization of fluid flow and pressure distributions using quiver and scatter plots.
3. Effective demonstration of how neural networks can integrate physical laws into their training process to enhance predictions.

Mathematical Calculations for Fluid Dynamics

1. Synthetic Data Generation (Fluid Dynamics)

- generate synthetic data for velocity (u, v) and pressure (p) fields using simple functions.

- **Velocity field equations**

  $u(x, y, t) = \sin(\pi * x) * \cos(\pi * y) * \exp(-t)$

  $v(x, y, t) = -\cos(\pi * x) * \sin(\pi * y) * \exp(-t)$

- **Pressure field equation:**

  $p(x, y) = \cos(\pi * x) * \cos(\pi * y)$

2. Neural Network Architecture for PINN

- The neural network takes three inputs (x, y, t) and predicts three outputs (velocity components u, v, and pressure p).
- The layers use tanh activation, common for approximating smooth functions like those in fluid dynamics.

3. Physics-Informed Loss Function (Navier-Stokes Equations)

The loss function is based on enforcing physical laws using the Navier-Stokes equations and continuity equation

- **Continuity equation (ensures incompressibility of fluid):**

  $\partial u/\partial x + \partial v/\partial y = 0$

- **Momentum equations (Navier-Stokes):**
  - **x-momentum equation**:

    $u\, \partial u/\partial x + v\, \partial u/\partial y + \partial p/\partial x - \nu\, (\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2) = 0$
  - **y-momentum equation**:

    $u\, \partial v/\partial x + v\, \partial v/\partial y + \partial p/\partial y - \nu\, (\partial^2 v/\partial x^2 + \partial^2 v/\partial y^2) = 0$

- **Data loss:**

  $\text{data\_loss} = \Sigma\, [(\text{u\_true} - \text{u\_pred})^2 + (\text{v\_true} - \text{v\_pred})^2 + (\text{p\_true} - \text{p\_pred})^2]$

- **Physics-based loss (enforcing physical constraints):**

  $\text{physics\_loss} = \Sigma\, [(\text{continuity\_loss})^2 + (\text{momentum\_x})^2 + (\text{momentum\_y})^2]$

- **Total loss**

  $\text{total\_loss} = \text{data\_loss} + \text{physics\_loss}$

4. Gradient Calculations

Using TensorFlow's `GradientTape`, calculate the following partial derivatives required for the

Navier-Stokes equations:

- $\partial u/\partial x$, $\partial u/\partial y$, $\partial v/\partial x$, $\partial v/\partial y$
- $\partial p/\partial x$, $\partial p/\partial y$