**Overview**

This code defines a neural network model (ShipSpeedPredictorModel) designed to predict a ship’s power consumption under various conditions (e.g., different speeds, drafts, wave heights, headings, etc.). The model isn’t just trained on direct power measurements (data-driven approach) but also incorporates physical knowledge in the form of PDE constraints, boundary conditions, and known physics-based relationships. This approach is often referred to as a Physics-Informed Neural Network (PINN).

**Inputs to the Neural Network**

**Model Input Features:**

The input features are a set of ship and environmental parameters. These can include (but are not limited to):

• **Speed-Through-Water:** The speed of the ship relative to the water.

• **Draft\_Fore and Draft\_Aft:** The fore and aft drafts of the ship (the vertical distance between the waterline and the bottom of the hull).

• **Significant\_Wave\_Height (H\_s):** The average height of the highest one-third of waves.

• **Mean\_Wave\_Direction and True\_Heading:** The direction from which waves are coming and the ship’s heading.

• Other environmental features such as wind speed and direction, and so forth (depending on the dataset).

**Data Scaling:**

Before feeding data into the network, the features are scaled (standardized) so that they have a mean of 0 and a standard deviation of 1. This helps the model train more efficiently.

**Outputs from the Neural Network**

**Predicted Power (P):**

The network outputs a single value: the ship’s predicted power consumption (scaled). After training, predictions are converted back to the original scale using the stored scaler. In other words, the neural network tries to learn a function:

**Model Architecture**

**Neural Network Structure:**

• The network is composed of multiple fully-connected (linear) layers with ReLU activations.

• Between layers, dropout is used to reduce overfitting.

• The final layer outputs a single value corresponding to the scaled power.

• Additionally, the model defines trainable physics-related parameters (e.g., k\_wave, k\_aw, k\_appendage, eta\_D, C\_d, and k\_trim). These parameters are part of the physics model and can be adjusted by training to better fit the observed data while still adhering to physical principles.

**Custom Weight Initialization:**

The model’s weights are initialized using Kaiming uniform initialization to promote stable training and reproducibility.

**Training: Data, PDE, Physics, and Boundary Losses**

The training process involves several types of losses that the network tries to minimize simultaneously. By combining these losses, we incorporate both data-driven accuracy and physical consistency.

1. **Data Loss (data\_loss):**

• **What it is:** The mean squared error (MSE) or mean absolute error (MAE) between the model’s predicted power and the actual measured power from the dataset.

• **Purpose:** Ensures the model predictions match the observed data closely.

Mathematically, if ytrue is the actual power and  ypred  is the predicted power:

A mathematical equation with numbers and symbols

Description automatically generated

2. **PDE Loss (pde\_loss):**

• **What it is:** The model uses a PDE residual that represents a physical relationship between power, speed, and their derivatives. We sample “collocation points” (sets of input features within the range of the training data) and enforce that the PDE relationship holds at these points.

• **Purpose:** Ensures the solution not only fits data points but also respects the underlying physical laws represented by a PDE. In other words, the PDE loss penalizes predictions that do not satisfy the physics-based differential equation constraints.

Conceptually:

• The PDE involves derivatives of the output (power) w.r.t. certain input variables (like speed).

• Let (N) denotes the PDE operator, the PDE residual N(ypred)

• To quantify how far the predictions are from satisfying the PDE, we calculate the **Mean Squared Error (MSE)** of the residuals:

A mathematical equation with numbers and symbols

Description automatically generated

• Here:

• M: The number of collocation points.

• N(ypred,j): The PDE residual at each collocation point.

3. **Physics Loss (physics\_loss):**

• **What it is:** A direct enforcement of known physical relationships for power and resistance. The code computes a physically-derived power (PS) from the input conditions (speed, wave height, etc.) using hydrodynamic equations and compares it with the network’s predicted power.

• **Purpose:** Encourages the model to produce predictions consistent with established physics. The model must learn parameters and a function that make its predictions align with known resistance and power formulas.

If PS is the physically computed power and Ppred the predicted power:

A mathematical equation with numbers and symbols

Description automatically generated

4. **Boundary Loss (boundary\_loss):**

• **What it is:** At certain boundary conditions (e.g., when the speed is zero), the power must also be zero. We generate “boundary points” (inputs that represent boundary conditions) and ensure the predicted power at these points adheres to the known boundary conditions.

• **Purpose:** Forces the model to respect known boundary conditions of the physical system.

For a boundary condition like if V=0 then P=0:

A mathematical equation with numbers and symbols

Description automatically generated

Boundary points NB are specific inputs (e.g., speed V = 0) that represent the boundary conditions of the system.

**Combining the Losses**

The total loss is a weighted combination of all these terms:



data\_loss\_coeff: The weight assigned to the **data loss** component.

physics\_loss\_coeff: The weight assigned to the **physics loss** component.

Pde\_loss\_coeff: weight assigned to the **PDE loss** component.

Pde\_loss\_coeff: The weight assigned to the **boundary loss** component.

**Data Flow**

1. **Training Data:**

Real measured data (features and power) is passed into the model to compute data\_loss.

2. **Collocation Points (for PDE Loss):**

Synthetic input points within the data’s range are generated to enforce PDE constraints (no direct measured power needed here).

3. **Boundary Points (for Boundary Loss):**

Synthetic input points representing boundary conditions (e.g., zero speed) are generated to ensure correct behavior at boundaries.

4. **Physics Computations (for Physics Loss):**

Using the input features from the training data, the code computes a physical estimate of power and compares it with the model’s predictions.

**Validation and Testing**

The code splits data into training, validation, and test sets. After training, it evaluates on validation and test sets to ensure the model generalizes well. It also conducts cross-validation and hyperparameter search to find the best learning rate and batch size.