Beyond Trial and Error: A Machine Learning Approach to Optimal Centrifugal Pump Impeller Trimming

A Genetic Algorithm Hyperparameter Optimization Approach

Mohamed Farid Khalil[a,1,*], Mohammed Twheed Khater[a,2], Seif Ibrahim Hassan[a,3]

[a] *Alexandria University, Mechanical Engineering, Alexandria, Egypt,*

Abstract

This paper proposes a genetic algorithm (GA)-based methodology to optimize hyperparameters for artificial neural networks (NNs) applied to pump impeller trimming. Traditionally, impeller trimming relies on expert knowledge and empirical methods, leading to time-consuming and potentially suboptimal outcomes. This work introduces a data-driven approach using NNs trained to predict the impeller diameter (D) and pump power (P), which can be calculated from flow rate (Q), density (ρ), head (H), and efficiency (η) using the equation P = (Q * ρ * g * H) / η. based on the desired operating point (flow rate (Q) and head (H)). A GA is employed to identify the optimal NN hyperparameters, including hidden layer size, training function, activation function, and maximum epochs. The goal is to minimize the mean squared error (MSE) between the network's predictions and the actual performance data. The paper details the implementation of the GA optimization process and discusses the key components and their significance in achieving optimal impeller trimming through NN predictions.

Keywords: Pump impeller trimming, Neural networks, Hyperparameter optimization, Genetic algorithms, Mean squared error

*Corresponding author

Email addresses: mfaridkhalil@yahoo.com (Mohamed Farid Khalil), mohammedtwheed@gmail.com (Mohammed Twheed Khater)

[1] Professor, Mechanical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt.

[2] Undergraduate, Mechanical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt.

[3] Undergraduate, Mechanical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt.

Nomenclature

NN	Neural Network
GA	Genetic Algorithm
MSE	Mean Square Erorr
NNs	Neural Networks
CNNs	Convolution Neural Networks
RNNs	Recurrent Neural Networks

1. Introduction

Pump impeller trimming is a crucial procedure in optimizing the performance of centrifugal pumps for specific operating conditions. It involves modifying the impeller geometry, typically by removing material from the blades, to achieve desired hydraulic characteristics such as flow rate (Q), head (H), and efficiency (η). Traditionally, this process has relied heavily on **empirical methods and engineering expertise**. These methods can be time-consuming, require significant trial-and-error experimentation, and may not always achieve the optimal trimming configuration.

The introduction of artificial neural networks (NNs) offers a promising **data-driven** alternative for automating and enhancing the pump impeller trimming process. NNs excel at modeling complex relationships between input data and desired outputs. By training an NN on a dataset of impeller designs and their corresponding performance outcomes (Q, H, η), the network can learn to predict the performance of new impellers based solely on their geometries.

In the context of pump impeller trimming, the input data to the NN would typically consist of the desired operating point of the pump, specified by the flow rate (Q) and head (H). The target data, or the network's output, would be the impeller's outer diameter (D) and the pump efficiency (η). However, directly predicting η is challenging due to its dependence on multiple factors beyond impeller geometry. Therefore, this work proposes using the NN to predict the impeller diameter (D) as the primary design variable. The pump efficiency (η) can then be calculated indirectly using the equation:

$$P = \frac{Q * \rho * g * H}{\eta}$$

where:

- P is the pump power (kW)
- ρ is the fluid density (kg/m^3)
- g is the acceleration due to gravity (m/s^2)

This approach allows the NN to focus on predicting the impeller geometry (D) that directly influences the pump's hydraulic performance, while the efficiency can be estimated based on the predicted D and the operating conditions (Q, H).

Achieving optimal NN performance for impeller trimming prediction requires careful selection of **hyperparameters**. These parameters define the network architecture and learning process, and significantly influence the NN's ability to learn complex relationships from the data. Key NN hyperparameters include:

- **Hidden layer size:** The number of neurons in the hidden layer(s) determines the network's capacity to learn complex patterns.
- **Training function:** This function dictates how the network updates its internal weights during the training process to minimize the error between predictions and actual outputs.
- **Activation function:** This function introduces non-linearity into the network, allowing it to model more complex relationships than a linear model.
- **Maximum epochs:** This parameter specifies the number of training iterations the network undergoes before stopping.

Selecting the optimal hyperparameters for a specific task can be a complex process. This paper proposes a **genetic algorithm (GA)**-based approach to automate the hyperparameter optimization process for the NN used in pump impeller trimming prediction.

1.1. Methodology

This section outlines the methodology employed to optimize the hyperparameters of an artificial neural network (NN) for predicting pump impeller diameter (D) in the context of impeller trimming. The optimization process utilizes a genetic algorithm (GA) implemented using the MATLAB GA toolbox.

Data Acquisition and Preprocessing:

The methodology leverages a dataset containing impeller geometries and their corresponding performance data, including flow rate (Q), head (H), diameter (D), and pump power (P). While the dataset doesn't include a direct measurement of efficiency (η), it can be calculated using the provided information and the following equation:

$$P = \frac{Q * \rho * g * H}{\eta}$$

where:

- η is the pump efficiency (-)
- ρ is the fluid density (kg/m^3)
- g is the acceleration due to gravity (m/s^2)

This data can be obtained from various sources, such as experimental measurements, computational fluid dynamics (CFD) simulations, or a combination of both. **Our** code (refer to the provided MATLAB code for specific details)[4] performs the following data preprocessing steps:

1. **Data Cleaning:** The data is inspected for missing values or outliers. Missing values can be addressed through techniques like imputation or deletion, while outliers may require further investigation or removal if they significantly impact the training process.
2. **Normalization:** The data is normalized to a specific range (often -1 to 1 or 0 to 1) using techniques like `mapminmax` in MATLAB. This helps ensure the gradients calculated during backpropagation have a more consistent magnitude, facilitating faster convergence and avoiding local minima during training.

Neural Network Architecture:

The proposed NN architecture utilizes a supervised learning approach for regression. The specific details in the code will determine the exact structure, but a typical configuration might involve:

- **Input Layer:** This layer consists of two neurons, corresponding to the input features: flow rate (Q) and head (H).
- **Hidden Layer(s):** One or more hidden layers are employed to learn complex relationships between the input and output data. The GA will optimize the number of neurons in the hidden layer(s) based on performance.
- **Output Layer:** The output layer contains a single neuron responsible for predicting the impeller diameter (D).

The activation functions used in each layer will also be optimized by the GA. The code specifically uses two common choices for activation functions in regression tasks:

[4] github link to our code:https://github.com/MohammedTwheed/trimming-code

- **Sigmoid (Logistic Function):** This function takes the form `f(x) = 1 / (1 + exp(-x))`. It squashes the input values between 0 and 1, introducing non-linearity into the network. Mathematically, the sigmoid function can be expressed as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- **Tanh (Hyperbolic Tangent Function):** This function takes the form `f(x) = (tanh(x)) = (e^x - e^{-x}) / (e^x + e^{-x})`. It squashes the input values between -1 and 1, providing a wider range of output compared to the sigmoid function. Mathematically, the tanh function can be expressed as:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Both sigmoid and tansig (which is mathematically equivalent to tanh) functions introduce non-linearity into the network, allowing it to model complex relationships between the input and output data. The GA will evaluate the performance of the NN using different activation functions (sigmoid and tansig) and select the one that leads to the minimal MSE between predicted and actual data.

The tansig function used in your code is defined as:

$$f(x) = tansig(x) = \frac{2}{1 + exp(-2x)} - 1$$

Genetic Algorithm (GA) for Hyperparameter Optimization:

The GA serves as the core optimization engine, searching for the optimal combination of NN hyperparameters that minimizes the mean squared error (MSE) between the network's predicted diameter (D) and the actual values in the training data. and here is some glimpses on how in general it works:

1. **Population Initialization:** The GA starts with a population of individuals (candidate hyperparameter sets). Each individual represents a unique configuration for the NN, including hidden layer size, training function, activation function, and maximum epochs.
2. **Fitness Evaluation:** Each individual in the population undergoes evaluation. The code trains an NN using the specific hyperparameters encoded in the individual's chromosome. The resulting NN's performance is then measured by calculating the MSE between its predicted diameter (D) and the actual values on a validation dataset (a portion of the original data set aside for evaluation).
3. **Selection:** Based on the fitness values (MSE), the GA selects a subset of individuals with superior performance (low MSE) to become parents for the next generation.
4. **Crossover:** Selected parent individuals undergo crossover, where portions of their genetic material (hyperparameter configurations) are exchanged to create new offspring (candidate solutions).

5. **Mutation:** A small probability of mutation is introduced to introduce random variations in the offspring's chromosomes. This helps maintain genetic diversity and explore new regions of the search space.
6. **Replacement and Termination:** The new generation of offspring replaces a portion of the lower-performing individuals in the population. The GA iterates through these steps until a stopping criterion is met, such as reaching a maximum number of generations or achieving a desired level of convergence (minimum MSE).

The code (refer to the MATLAB code for specific details) will implement the specific selection, crossover, and mutation operators used in the GA and all of this is handled by `ga` the MATLAB toolbox.

Training and Validation:

The final, selected hyperparameter configuration from the GA is used to train a new NN model. This model is trained on a separate training dataset, excluding the data used for validation during the GA optimization process. The trained model is then evaluated on a hold-out test dataset to assess its generalizability and prediction accuracy on unseen data.

Performance Evaluation:

The performance of the trained NN model is evaluated based on various metrics, including:

- **Mean Squared Error (MSE):** This metric measures the average squared difference between the predicted impeller diameters (D) and the actual values in the test dataset. A lower MSE indicates better prediction accuracy.
- **Coefficient of Determination (R-squared):** This metric indicates the proportion of the variance in the actual diameter data explained by the NN model's predictions. A value closer to 1 signifies a stronger correlation between the predicted and actual values.
- **Visualization Techniques:** Visualizations such as scatter plots comparing predicted vs. actual data.

2. Code Documentation

1. loadData function:

- This function loads data from four separate `.mat` files containing flow rate (Q), head (H), diameter (D), and power (P) values.
- It performs error handling to ensure the data path exists and loads the data into MATLAB variables using `load(fullfile(dataPath, 'QH.mat'))`.

- The function uses `transpose` on loaded variables (`QH.QH`, `D.D`, etc.) to convert them from row vectors to column vectors. This is likely because the `train` function in MATLAB expects training data as column vectors (`number of inputs x number of examples`).

2. optimizeNNForTrimmingPumpImpeller function:

- This core function performs the optimization for a neural network using a genetic algorithm (GA).
- It takes input data (`x` - flow rate, head), target data (`t` - diameter or power), and an optional user-specified random seed (`userSeed`) for reproducibility.
- It defines several key elements:
  - **Training Function Options:** A list of possible training functions like `'trainlm'`, `'trainbr'`, etc. These functions determine how the network weights are updated during training.
  - **Activation Function Options:** A list of possible activation functions like `'tansig'`, `'logsig'`. These functions introduce non-linearity into the network's behavior.
  - **Bounds for Hyperparameters:** Lower and upper bounds for the number of hidden layer neurons, epochs, training function index, and activation function index. These bounds are used by the GA for exploration.
  - **GA Options:** These options define the parameters for the GA, such as `MaxTime` (maximum optimization time) and potential stopping criteria.
  - **Global Variable:** `bestTrainedNet` is a global variable used to store the best trained neural network found during optimization.
- An important part of this function is the `evaluateHyperparameters` function (defined locally within `optimizeNNForTrimmingPumpImpeller`).
  - It takes hyperparameters (hidden layer size, epochs, training function index, activation function index) and the data (`x`, `t`) as input.
  - It builds a neural network based on the provided hyperparameters.
  - It trains the network using the training function and activation function specified by the indices.
  - It splits the data into training, validation, and testing sets.
  - It trains the network and calculates the mean squared error (MSE) on the validation set.
  - It returns the average MSE of the training, validation, and testing sets.
  - It also checks if the current MSE is the best so far and updates the `bestTrainedNet` global variable if necessary.
- The `optimizeNNForTrimmingPumpImpeller` function uses the GA to search for the combination of hyperparameters that minimizes the MSE returned by the `evaluateHyperparameters` function.
- It logs the optimization results to a file named `optimizeNNForTrimmingPumpImpeller_log.txt`, including optimized hyperparameters, final MSE, random seed, and optimization duration.

3. processDataAndVisualize function:

- This function processes the data and visualizes the results.
- It takes the pre-loaded data (`QH`, `D`, `QD`, `P`), the trained neural networks (`bestTrainedNetD` for diameter, `bestTrainedNetP` for power), and an optional `saveFigures` argument.
- It performs data interpolation using `griddata` to create smoother surfaces for visualization of predicted diameters and power.
- It then creates visualizations using `mesh` plots for both diameters and power, along with scatter plots of the original data points for comparison.
- The function utilizes transpose (`'`) again before feeding data to the neural networks with `sim(bestTrainedNetD, QH')` and `sim(bestTrainedNetP, QD')`. This is because the neural network expects the input data in transposed form (column vectors).
- It saves the visualizations as `.fig` and `.png` files if the `saveFigures` argument is set to `true`.

Overall Structure:

- The main script (`MAIN`) starts by loading the data using `loadData`.
- It then optimizes two separate neural networks:
  - One for predicting diameter (D) based on flow rate (Q) and head (H).
  - Another for predicting power consumption (P) based on flow rate (Q) and diameter (D).
- then plot the data against the neural network.

here is example plot for our data[5] Figure 1

References

[1] M. Khalil, M. Elgohary, A. Shaito, Application of artificial neural network for the prediction of trimmed impeller size of a centrifugal pump.

[5] we will find the dataset we used at: https://github.com/MohammedTwheed/trimming-code/tree/main/training-data

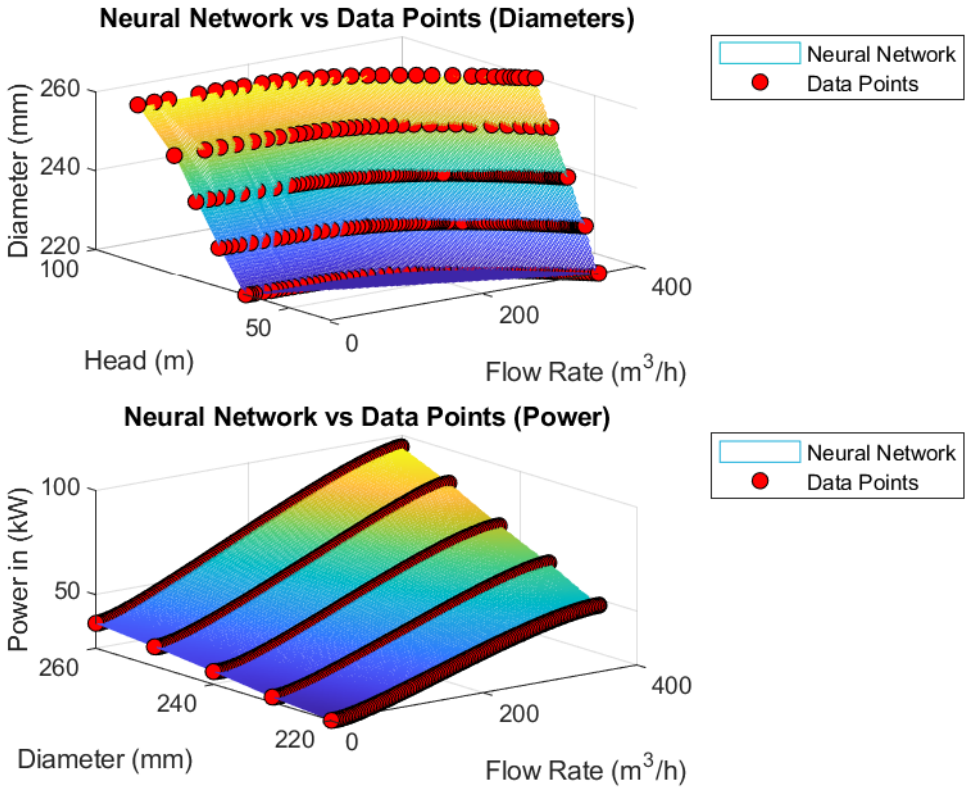

Figure 1: NN vs Data