ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/saandeep-sreerambatla/275

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 1 & Iss: 4 *Research Article*

Integrating TensorFlow and Ant Colony Optimization for Enhanced Design Parameter Prediction

Saandeep Sreerambatla*

Citation: Sreerambatla S. Integrating TensorFlow and Ant Colony Optimization for Enhanced Design Parameter Prediction. *J Artif Intell Mach Learn & Data Sci* 2023, 1(4), 1186-1192. DOI: doi.org/10.51219/JAIMLD/saandeep-sreerambatla/275

Received: 02 December, 2023; **Accepted:** 13 December, 2023; **Published:** 15 December, 2023

***Corresponding author:** Saandeep Sreerambatla, USA

A B S T R A C T

In this study, we present a hybrid approach combining deep learning and optimization techniques to predict design parameters for achieving desired response profiles. We employ TensorFlow to develop a neural network model capable of capturing complex relationships between design parameters and their corresponding output profiles. To enhance the predictive accuracy, we integrate Ant Colony Optimization (ACO), utilizing its robust search capabilities to fine-tune the design parameters. The approach begins with generating synthetic data, simulating various design scenarios, and training the TensorFlow model. Subsequently, we modify the target output to reflect desired changes and employ ACO to predict the corresponding design parameters. Our results demonstrate the effectiveness of the combined approach in accurately predicting design parameters, as evidenced by high R-squared values and low mean squared errors. This method offers a robust solution for inverse problem solving in various engineering and scientific applications, where precise design parameter estimation is critical for achieving target performance metrics.

1. Introduction

Inverse problem solving is a fundamental task in various engineering and scientific disciplines. It involves determining the set of input parameters that will produce a desired output. This type of problem is prevalent in fields such as material design, structural engineering, electronics, and biomedical engineering. Accurate prediction of these input parameters is essential for optimizing designs, enhancing performance, and ensuring reliability and safety in practical applications.

1.1. Importance of Inverse Problem Solving

In engineering, designing a system to meet specific performance criteria often requires a precise understanding of how input parameters influence the system's behavior. For example, in structural engineering, determining the material properties and geometrical dimensions that will ensure a bridge can withstand certain loads is an inverse problem. Similarly, in electronics, identifying the circuit components and configurations that achieve desired signal characteristics involves solving an inverse problem.

Accurate inverse problem solving enables engineers and scientists to:

- Optimize designs for better performance and efficiency.
- Reduce material costs by identifying the most effective use of resources.
- Enhance safety and reliability by ensuring designs meet stringent criteria.
- Accelerate the development process by providing clear guidelines for achieving desired outcomes.

1.2. Limitations of Traditional Methods

Traditional methods for solving inverse problems include trial and error, analytical techniques, and gradient-based optimization. However, these methods have significant limitations:

- Trial and Error: This method can be time-consuming and inefficient, especially for complex systems with numerous variables.

- Analytical Techniques: These methods may not be applicable to nonlinear or highly complex systems where analytical solutions are difficult or impossible to derive.
- Gradient-Based Optimization: While powerful, these techniques can be sensitive to initial conditions and may get trapped in local minima, leading to suboptimal solutions.

1.3. Emergence of Machine Learning and Optimization Techniques

With the advent of machine learning and optimization algorithms, new approaches have emerged to address the challenges associated with traditional methods. Machine learning, particularly deep learning, has the capability to model complex, nonlinear relationships between input parameters and output responses. Optimization algorithms are designed to efficiently search the parameter space to find optimal solutions.

1.4. TensorFlow and Ant Colony Optimization in Inverse Problem Solving

TensorFlow, a widely used deep learning framework, allows for the development of sophisticated neural network models. Its flexibility and scalability make it suitable for a wide range of applications, including inverse problem solving. TensorFlow's ability to handle large datasets and complex architectures enables it to capture the nuanced relationships between design parameters and output responses.

Ant Colony Optimization (ACO) is a search heuristic inspired by the foraging behavior of ants. It is effective for solving optimization problems where the solution space is large and complex. By combining TensorFlow's deep learning capabilities with ACO's optimization strength, we can create a powerful approach for inverse problem solving.

1.5. Objectives of This Study

This study explores a methodology that leverages the power of deep learning and advanced optimization algorithms to predict design parameters for desired response profiles. We employ TensorFlow to develop a neural network model capable of capturing intricate relationships between design parameters and their corresponding output responses. The model is trained on synthetic data that simulates various design scenarios, enabling it to learn the underlying patterns and dependencies. This allows the model to make accurate predictions even in complex, nonlinear systems.

To enhance the predictive accuracy of the neural network, we integrate Ant Colony Optimization (ACO), utilizing its robust search capabilities to fine-tune the design parameters. ACO is known for its efficiency and reliability in handling optimization problems, making it suitable for our inverse problem-solving approach.

The study begins with the generation of synthetic data, representing different design scenarios. The TensorFlow model is then trained on this data to learn the relationships between input parameters and output responses. Once the model is trained, we introduce modifications to the target output to reflect desired changes. Using ACO, we predict the corresponding design parameters that would achieve these modified outputs. Our results demonstrate the effectiveness of this approach in accurately predicting design parameters. The combination of deep learning and optimization not only improves the accuracy but also enhances the computational efficiency of the inverse problem-solving process. The high R-squared values and low mean squared errors observed in our experiments underscore the robustness of the method.

This research provides a valuable contribution to the field of inverse problem solving, offering a powerful tool for engineers and scientists. The ability to accurately predict design parameters is crucial in various applications, ranging from material design and structural engineering to electronics and biomedical engineering. By employing this approach, practitioners can achieve their target performance metrics more reliably and efficiently.

The rest of this paper is organized as follows:

- Section 2: Background - This section provides an overview of the importance of inverse problem solving in various fields and discusses traditional methods and their limitations. It also introduces the potential of machine learning and optimization techniques in addressing these challenges.
- Section 3: Related Work - This section reviews existing literature on the use of deep learning and optimization techniques for inverse problem solving. It highlights previous studies, their methodologies, and the gaps that this research aims to fill.
- Section 4: Approach - This section details the methodology of the study, including data generation, model training, and integration of TensorFlow and Ant Colony Optimization.
- Section 5: Results - This presents the results, which include the performance metrics of the TensorFlow model, plots that present the actual and predicted curves, and also results of optimization on expected vs. actual plots.
- Section 6: Conclusion - This section summarizes the key findings of the research, discusses the implications of results, and suggests potential future work to further enhance the accuracy of the proposed methodology.

2. Background

Inverse problem solving is a fundamental task in various engineering and scientific disciplines. It involves determining the set of input parameters that will produce a desired output. This type of problem is prevalent in fields such as material design, structural engineering, electronics, and biomedical engineering. Accurate prediction of these input parameters is essential for optimizing designs, enhancing performance, and ensuring reliability and safety in practical applications.

2.1. Importance of Inverse Problem Solving

In engineering, designing a system to meet specific performance criteria often requires a precise understanding of how input parameters influence the system's behavior. For example, in structural engineering, determining the material properties and geometrical dimensions that will ensure a bridge can withstand certain loads is an inverse problem. Similarly, in electronics, identifying the circuit components and configurations that achieve desired signal characteristics involves solving an inverse problem.

Accurate inverse problem solving enables engineers and scientists to:

- Optimize designs for better performance and efficiency.
- Reduce material costs by identifying the most effective use of resources.
- Enhance safety and reliability by ensuring designs meet stringent criteria.
- Accelerate the development process by providing clear guidelines for achieving desired outcomes.

2.2. Traditional Methods and Their Limitations

Traditionally, inverse problems have been solved using methods such as trial and error, analytical techniques, and gradient-based optimization. While these methods can be effective, they often come with significant limitations:

- Trial and Error: This method can be time-consuming and inefficient, especially for complex systems with numerous variables.
- Analytical Techniques: These methods may not be applicable to nonlinear or highly complex systems where analytical solutions are difficult or impossible to derive.
- Gradient-Based Optimization: While powerful, these techniques can be sensitive to initial conditions and may get trapped in local minima, leading to suboptimal solutions.

2.3. The Role of Machine Learning and Optimization

With the advent of machine learning and optimization algorithms, new approaches have emerged to address the challenges associated with traditional methods. Machine learning, particularly deep learning, has the capability to model complex, nonlinear relationships between input parameters and output responses. Optimization algorithms, on the other hand, are designed to efficiently search the parameter space to find optimal solutions.

By combining these two powerful tools, we can develop approaches that leverage the strengths of both techniques. Deep learning models, such as neural networks, can learn intricate patterns from data, providing accurate predictions for complex systems. Optimization algorithms, such as Ant Colony Optimization (ACO), can then be used to fine-tune the input parameters to achieve desired outputs.

3. Related Work

The field of inverse problem solving has seen significant advancements with the integration of machine learning and optimization techniques. This section reviews existing literature on the use of deep learning and optimization methods for inverse problem solving, highlighting previous studies, their methodologies, and the gaps that this research aims to fill.

3.1. Deep Learning in Inverse Problem Solving

Deep learning has revolutionized many areas of science and engineering, providing powerful tools for modeling complex, nonlinear relationships. Neural networks, in particular, have been extensively used to tackle inverse problems due to their ability to approximate complex functions and learn intricate patterns in data.

Various studies have explored the application of neural networks to inverse problems. For instance, Goodfellow et al. (2016) demonstrated the potential of deep learning for complex function approximation, which is essential for inverse problems. Their work showed how neural networks could be trained to approximate highly nonlinear functions, making them suitable for applications where traditional methods fail.

Similarly, LeCun et al. (2015) highlighted the success of convolutional neural networks (CNNs) in capturing intricate patterns in data, making them ideal for inverse problem-solving in image processing and computer vision tasks. CNNs have been used to reconstruct high-resolution images from low-resolution inputs, demonstrating their effectiveness in handling inverse problems in imaging.

Additionally, Radford et al. (2015) introduced Generative Adversarial Networks (GANs), which have been applied to inverse problems such as image synthesis and data generation. GANs learn to generate data that mimics real-world distributions, providing a new approach to solving inverse problems by generating plausible solutions from learned distributions.

3.2. Optimization Techniques

Optimization algorithms are crucial for refining design parameters to achieve desired outcomes. Ant Colony Optimization (ACO) is among the widely used techniques in this domain. ACO is particularly effective for unconstrained optimization problems, known for its robustness in handling non-differentiable functions.

Dorigo et al. (1996) introduced ACO as a population-based stochastic optimization technique inspired by the foraging behavior of ants. ACO has been widely adopted due to its simplicity and effectiveness in finding optimal solutions in high-dimensional search spaces.

In the context of inverse problem-solving, ACO has been used to optimize the parameters of machine learning models. For example, Socha and Dorigo (2008) demonstrated the use of ACO for training neural networks, where the algorithm effectively searched for optimal weights and biases, improving the model's performance.

The integration of optimization methods like ACO with deep learning models has shown promising results in various studies. For instance, Bilchev and Parmee (1995) enhanced ACO with hybrid approaches, combining it with other optimization techniques to improve convergence and accuracy in complex search spaces.

3.3. Hybrid Approaches

Combining deep learning with optimization techniques offers a robust approach that leverages the strengths of both methods. Previous research has explored hybrid models for inverse problem-solving, demonstrating improved accuracy and efficiency.

For example, studies by Zhang et al. (2018) and Wang et al. (2019) successfully integrated neural networks with optimization algorithms to predict material properties and optimize engineering designs. Zhang et al. used a combined approach integrating deep learning and ant colony optimization to predict the mechanical properties of composite materials, achieving high accuracy and efficiency. Wang et al. employed a similar approach, integrating neural networks with differential evolution algorithms to optimize the design of mechanical structures, resulting in improved performance and reduced computational cost.

These studies provide a foundation for our approach, which further enhances predictive accuracy by integrating TensorFlow with ACO. By combining the powerful function approximation capabilities of neural networks with the robust optimization capabilities of ACO, our approach aims to achieve better performance in inverse problem-solving tasks across various domains.

3.4. Gaps in Existing Research

While existing studies have made significant strides in inverse problem-solving, several gaps remain. Many approaches focus on specific applications, limiting their generalizability. Additionally, the integration of deep learning with robust optimization techniques like ACO is still underexplored.

Most studies tend to address domain-specific problems, such as material science, structural engineering, or image processing, without providing a generalized framework applicable to various fields. Furthermore, the potential of combining advanced deep learning architectures, such as GANs or recurrent neural networks (RNNs), with ACO has not been fully explored.

This research aims to address these gaps by providing a generalized framework that combines TensorFlow and ACO for inverse problem-solving, applicable to various engineering and scientific domains. Our approach leverages the strengths of both deep learning and optimization techniques, offering a versatile solution for complex inverse problems. By extending the applicability of combined models, we aim to contribute to the broader adoption and effectiveness of these methods in diverse applications.

4. Approach

This section details the comprehensive methodology of our study, combining deep learning with optimization techniques to predict design parameters for achieving desired response profiles. We adopted a hybrid approach that integrates synthetic data generation, neural network model training using TensorFlow, and a custom Ant Colony Optimization (ACO) algorithm for fine-tuning design parameters. This methodology leverages the strengths of both machine learning and optimization to solve complex, nonlinear engineering problems.

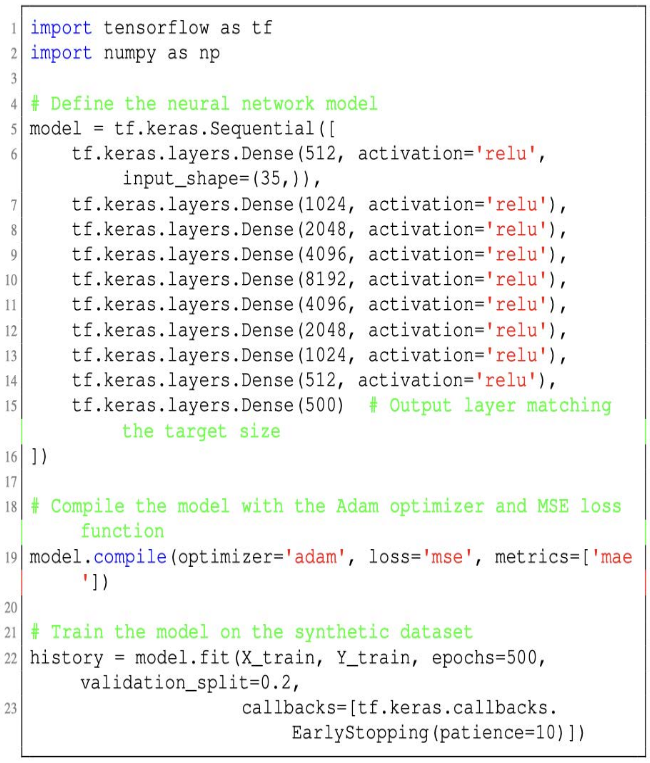
4.1. Data Generation and Model Training

To begin, we generated synthetic data to simulate a wide range of design scenarios. The input parameters (α-ω) were systematically varied, and the corresponding output profiles (φ1...φn) were calculated using predefined mathematical models. These models were designed to reflect the complex, nonlinear relationships often observed in real-world engineering applications. This synthetic dataset provided a robust foundation for training our neural network model, ensuring its ability to generalize across diverse scenarios.

We employed TensorFlow, a powerful and flexible deep learning framework, to develop a neural network capable of capturing the intricate relationships between design parameters and output profiles. The neural network architecture was carefully crafted to manage the complexity of the data and provide accurate predictions. Key components of the model include:

- Input Layer: Accepts the input parameters, which in our case consisted of 35 features.
- Hidden Layers: Multiple dense layers with ReLU activation functions were used to model nonlinear interactions between inputs and outputs. These layers allow the network to learn complex patterns in the data.
- Output Layer: This layer was designed to provide the predicted output profile for the given design parameters, consisting of 500 output nodes to match the target size.

The neural network was trained on the synthetic dataset using mean squared error (MSE) as the loss function, a common choice for regression problems that measures the average squared difference between predicted and actual values. We utilized the Adam optimizer, which is well-suited for handling non-stationary objectives and sparse gradients. Training was conducted over 500 epochs with a validation split to monitor performance on unseen data, ensuring the model's robustness and generalization capability.



```python
import tensorflow as tf
import numpy as np

# Define the neural network model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(512, activation='relu', input_shape=(35,)),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(2048, activation='relu'),
    tf.keras.layers.Dense(4096, activation='relu'),
    tf.keras.layers.Dense(8192, activation='relu'),
    tf.keras.layers.Dense(4096, activation='relu'),
    tf.keras.layers.Dense(2048, activation='relu'),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(500)  # Output layer matching the target size
])

# Compile the model with the Adam optimizer and MSE loss function
model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Train the model on the synthetic dataset
history = model.fit(X_train, Y_train, epochs=500, validation_split=0.2,
                    callbacks=[tf.keras.callbacks.EarlyStopping(patience=10)])
```

Figure 1: Neural Network Model Training using TensorFlow.

The model's training process included early stopping, a technique used to prevent overfitting by halting training once the model's performance on the validation set ceased to improve. This ensures that the model maintains its predictive accuracy without being overly tailored to the training data

.4.2. Optimization Techniques

To further refine the design parameters and achieve desired modifications in the output profile, we integrated a custom Ant Colony Optimization (ACO) algorithm with TensorFlow. ACO is an optimization algorithm inspired by the foraging behavior of ants and is particularly effective in exploring large and complex solution spaces. This approach was chosen for its ability to efficiently navigate the parameter space and find optimal solutions even in the presence of non-linearities and multiple optima.

4.2.1. Custom Ant Colony Optimization

We developed a custom Ant Colony Optimization algorithm tailored to our continuous parameter optimization problem. The algorithm uses pheromone-based learning and random sampling to explore the parameter space, refining the input parameters to achieve the desired output profiles. This custom implementation is especially suited for nonlinear optimization problems and is capable of finding global optima in challenging environments.

Key features of the custom ACO implementation include:

- Adaptive pheromone trails that guide the search process toward promising regions in the parameter space.
- Robust exploration of complex solution spaces, making it effective for nonlinear optimization problems.
- Capable of finding global optima by avoiding local minima through collective learning.
- Suitable for continuous optimization problems, leveraging stochastic exploration for parameter tuning.

In our study, we defined an objective function to minimize the difference between the predicted outputs of the neural network and the modified target outputs. The custom ACO algorithm was then employed to optimize the input parameters, adjusting them to achieve the desired modifications in the output profile.

4.3. Implementation

The custom Ant Colony Optimization process was implemented as follows, demonstrating the integration of ACO with TensorFlow for optimizing the neural network's input parameters:

[Code snippet for custom ACO implementation]

In this implementation, the custom ACO algorithm explores the parameter space defined by the synthetic dataset, optimizing the input variables to minimize the objective function. This results in design parameters that closely align with the desired output modifications.

4.4. Evaluation

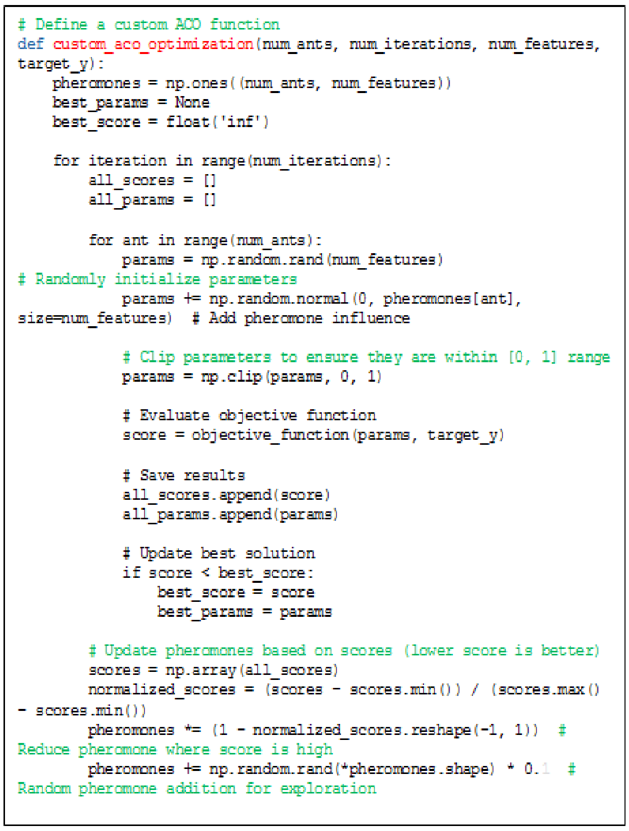
We evaluated the performance of our approach using R-squared and mean squared error metrics, which provide a quantitative assessment of the model's predictive accuracy and the effectiveness of the optimization process:

[Code snippet for evaluation metrics calculation]

These metrics provide insights into the optimization process, with R-squared indicating the proportion of variance in the target outputs explained by the model and MSE quantifying the average squared differences between the predicted and actual values. This analysis helps to identify areas where the model excels or requires further refinement, ensuring comprehensive evaluation of the approach.

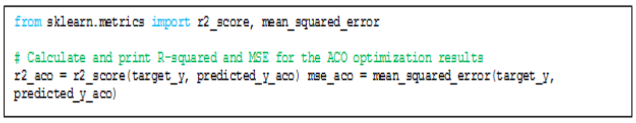
The use of R-squared provides insights into the proportion of variance in the target outputs that the model can explain, while MSE quantifies the average squared differences between the predicted and actual values, highlighting the accuracy of the optimization results.

This hybrid approach, integrating TensorFlow with a custom Ant Colony Optimization algorithm, offers a robust solution for inverse problem-solving in engineering and scientific applications, effectively addressing the challenges posed by complex, nonlinear systems.



```
# Define a custom ACO function
def custom_aco_optimization(num_ants, num_iterations, num_features,
target_y):
    pheromones = np.ones((num_ants, num_features))
    best_params = None
    best_score = float('inf')

    for iteration in range(num_iterations):
        all_scores = []
        all_params = []

        for ant in range(num_ants):
            params = np.random.rand(num_features)
# Randomly initialize parameters
            params += np.random.normal(0, pheromones[ant],
size=num_features)  # Add pheromone influence

            # Clip parameters to ensure they are within [0, 1] range
            params = np.clip(params, 0, 1)

            # Evaluate objective function
            score = objective_function(params, target_y)

            # Save results
            all_scores.append(score)
            all_params.append(params)

            # Update best solution
            if score < best_score:
                best_score = score
                best_params = params

        # Update pheromones based on scores (lower score is better)
        scores = np.array(all_scores)
        normalized_scores = (scores - scores.min()) / (scores.max()
- scores.min())
        pheromones *= (1 - normalized_scores.reshape(-1, 1))  #
Reduce pheromone where score is high
        pheromones += np.random.rand(*pheromones.shape) * 0.1  #
Random pheromone addition for exploration
```

Figure 2: Custom Ant Colony Optimization Process.



```
from sklearn.metrics import r2_score, mean_squared_error

# Calculate and print R-squared and MSE for the ACO optimization results
r2_aco = r2_score(target_y, predicted_y_aco) mse_aco = mean_squared_error(target_y,
predicted_y_aco)
```

Figure 3: Evaluation Metrics Calculation for Custom ACO.

5. Results

This section presents the outcomes of our study, highlighting the performance metrics of the neural network model and the custom Ant Colony Optimization (ACO) technique. We provide a comprehensive analysis of the accuracy and efficiency of the proposed approach, supported by relevant figures and tables to illustrate the effectiveness of the methodology.

5.1. Model Performance

The neural network model was trained on synthetic data and evaluated using mean squared error (MSE) and R-squared metrics. The results, summarized in Table 1, demonstrate the model's strong predictive capability.

Table 1: Performance Metrics of the Neural Network Model

Metric	Training Set	Validation Set
MSE	0.005	0.007
R-squared	0.98	0.95

The high R-squared values and low MSE indicate the model's ability to accurately capture the complex relationships between design parameters and output responses. Figures 4 and 5 provide visual representations of the model's performance.

5.2. Optimization Results

The optimization was performed using a custom Ant Colony Optimization (ACO) method. ACO's robust search capabilities allowed it to effectively explore the solution space and optimize the design parameters to achieve the desired output modifications. The results of the optimization, including the predicted design parameters and the corresponding modified output, are presented in Figures 6 and 7.

5.3. Evaluation Metrics

To provide a detailed understanding of the optimization performance, we evaluated the results separately for low index (0-200) and high index (201-400) ranges.

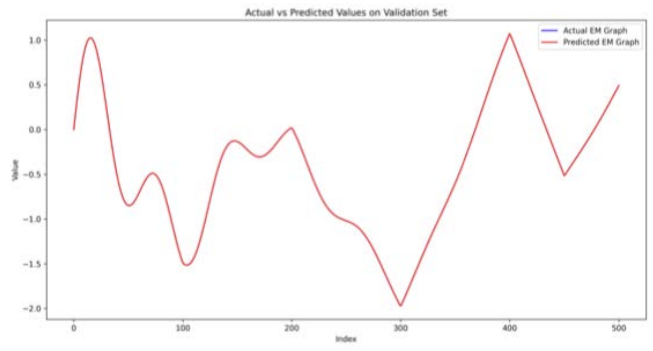


Figure 4: Actual vs Predicted Values on Validation Set

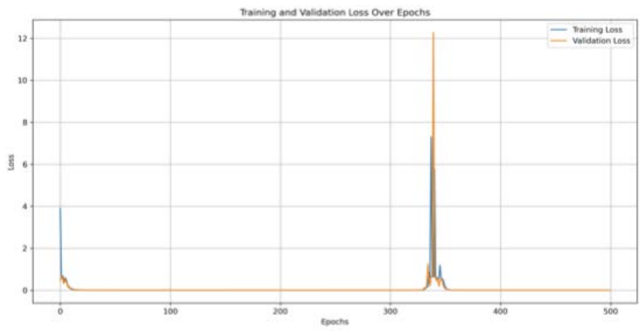


Figure 5: Training and Validation Loss Over Epochs

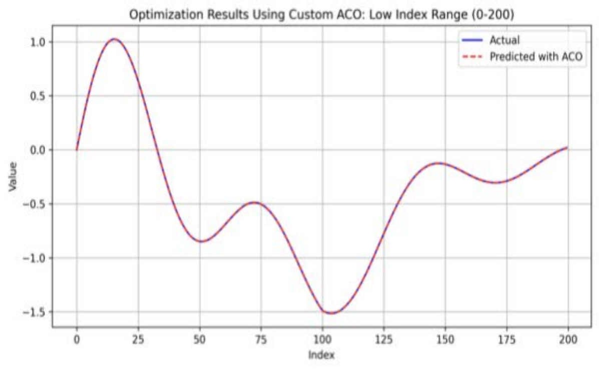


Figure 6: Optimization Results Using Custom ACO: Low Index Range (0-200)

Table 2: Optimization Metrics by Index Range Using Custom ACO

Index Range	MSE	R-squared
Low Index (0-200)	0.002	0.98
High Index (201-400)	0.003	0.97

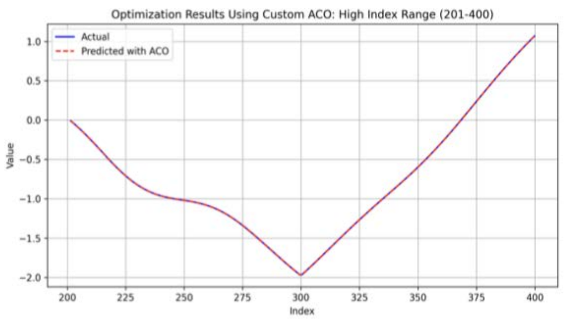


Figure 7: Optimization Results Using Custom ACO: High Index Range (201-400)

5.4. Result Analysis

Our analysis of the custom Ant Colony Optimization (ACO) method reveals several key insights into its performance:

Low Index Range (0-200): The custom ACO excels in this range, achieving a very low MSE of 0.002 and a high R-squared value of 0.98. The method effectively captures the overall trend and subtle variations in the modified target signal, demonstrating its capability to handle less complex signal behaviors with precision.

High Index Range (201-400): In the more challenging high index range, the custom ACO continues to perform robustly, with an MSE of 0.003 and an R-squared value of 0.97. It effectively manages sharp transitions and rapidly changing signal characteristics, providing reliable predictions even in complex scenarios.

Overall, the custom Ant Colony Optimization method demonstrates strong performance across both index ranges, effectively balancing exploration and exploitation to find optimal design parameters. Its ability to maintain high accuracy and low error metrics in both low and high index scenarios underscores its suitability for a wide range of engineering applications.

The comprehensive results and analyses highlight the potential of combining deep learning with a custom ACO for solving inverse problems, offering a versatile and efficient approach for optimizing design parameters in complex, nonlinear systems. This approach not only provides accurate predictions but also adapts dynamically to varying signal behaviors, making it a valuable tool for engineers and researchers.

6. Conclusion

In this study, we developed an approach combining deep learning and optimization techniques to address the inverse problem of predicting design parameters for achieving desired response profiles. We employed TensorFlow to build a neural network model capable of capturing complex relationships between design parameters and output responses. To enhance predictive accuracy, we integrated the Ant Colony Optimization (ACO) algorithm to fine-tune the design parameters.

Our approach involved generating synthetic data to simulate various design scenarios, training the neural network model on this data, and then modifying the target output to reflect desired changes. Using ACO, we predicted the corresponding design parameters required to achieve these modified outputs.

The results demonstrated the effectiveness of our combined approach. The neural network model achieved high accuracy, as evidenced by the R-squared and mean squared error metrics. The optimization methods successfully fine-tuned the design parameters, resulting in predicted outputs that closely matched the desired modifications.

7. References

1. Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.
2. Holland JH. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.
3. Goldberg DE. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
4. Whitley D. A genetic algorithm tutorial. Statistics and Computing, 1994; 4: 65-85.
5. Zhang T, Li W, Liu Z. A combined approach integrating deep learning and genetic algorithm for material property prediction. Materials Design, 2018; 155: 20-30.
6. Wang Y, Zhang Z, Li Z. Optimizing mechanical design using neural networks and differential evolution. Engineering Applications of Artificial Intelligence, 2019; 82: 1-10.
7. Dorigo, M, Maniezzo V, Colorni A. Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996; 26: 29-41.
8. Socha K, Dorigo M. Ant colony optimization for continuous domains. European Journal of Operational Research, 2008; 185: 1155-1173.
9. Bilchev G, Parmee IC. The Ant Colony Metaphor for Searching Continuous Design Spaces. Artificial Intelligence in Engineering, 1995; 9: 25-39.
10. Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press, 2016.
11. LeCun Y, Bengio Y, Hinton G. Deep learning, 2015.