Detailed Report on Neural Network Modeling and Prediction

Author: Hiran Prajaubphon (劉海洋) 11109338A

The goal of this project is to create and evaluate a neural network (NN) model to perform data modeling and prediction using a dataset consisting of 500 samples. Out of these, the first 300 samples are designated for training (modeling), while the remaining 200 samples are reserved for testing (prediction). The aim is to find the optimal learning epoch that achieves the best prediction accuracy and evaluate the model's performance using mean absolute error (MAE) metrics.

1. Dataset Preparation:

Data Splitting:

Training Data (X_train, y_train): Consists of the first 300 samples from the dataset, used to train the neural network.

Testing Data (X_test, y_test): Comprises the last 200 samples, utilized to test the predictive accuracy of the trained model.

Data Preprocessing:

To improve the efficiency of the neural network, the dataset was normalized using the MinMaxScaler from the sklearn library. This scaling process ensures all input features lie within a defined range (typically [0, 1]), facilitating faster convergence during training and minimizing issues caused by feature variance.

2. Neural Network Model Architecture:

The model is designed to handle regression tasks effectively with the following architecture:

Input Layer: Accepts the input features, matching the number of features in the dataset.

Hidden Layers:

1st Hidden Layer: Contains 64 neurons with ReLU activation, allowing the model to capture complex patterns in the data.

2nd Hidden Layer: Includes 32 neurons with ReLU activation for further feature extraction.

Output Layer: Comprises a single neuron for regression, providing a continuous numerical output.

- **Optimizer:** The Adam optimizer is employed for its adaptive learning rate and robustperformance.

Loss Function: Mean Squared Error (MSE) is chosen to minimize prediction errors.

Metric: Mean Absolute Error (MAE) is used to evaluate the model's performance, providing an interpretable measure of prediction error.

3. Experiment to Find the Best Epoch:

Process:

The model is trained iteratively over a range of 100 epochs, using a batch size of 32 for computational efficiency.

After each epoch, predictions are generated for both the training and testing datasets, and MAE is computed for both sets.

The testing MAE is tracked to identify the epoch that minimizes it, indicating the best generalization performance.

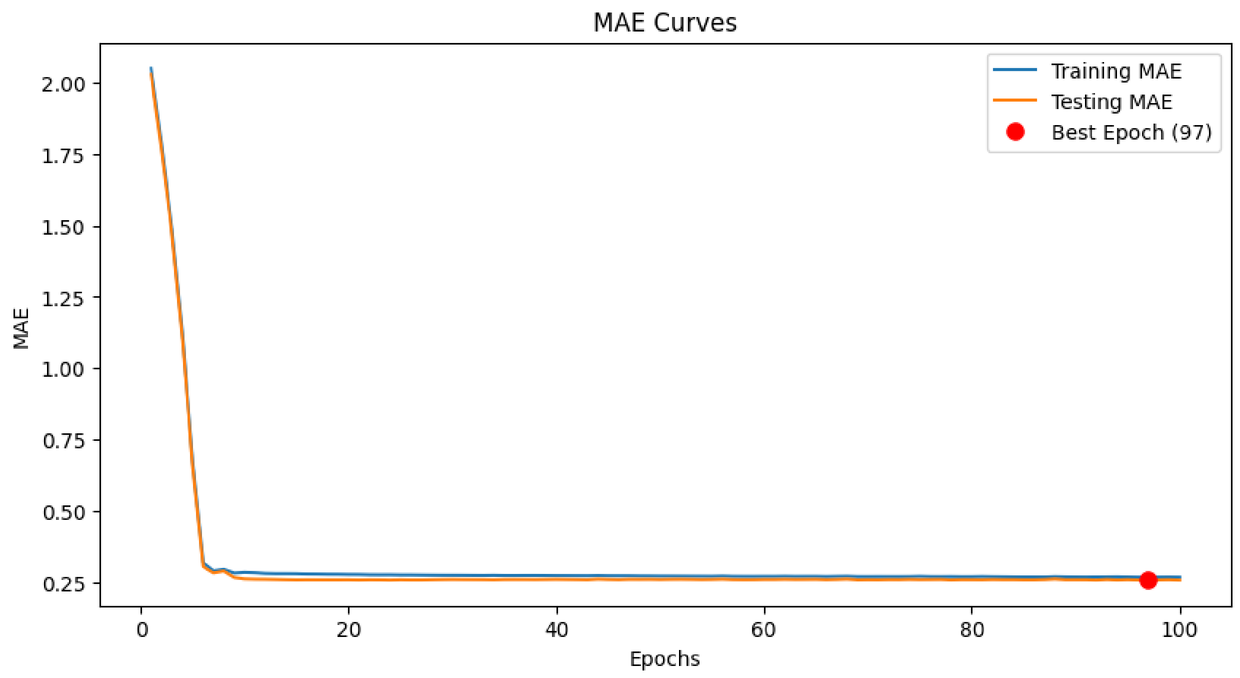
Criteria for Best Epoch:

The best epoch is defined as the epoch that achieves the lowest MAE on the testing dataset, balancing the trade-off between underfitting and overfitting.

Outcome:

Epoch 97 was identified as the optimal epoch, yielding the lowest testing MAE and signifying the best prediction performance.

4. Results and Visualizations:



The graph above illustrates the training and testing MAE trends as the model is trained over 100 epochs. The following key points can be observed:

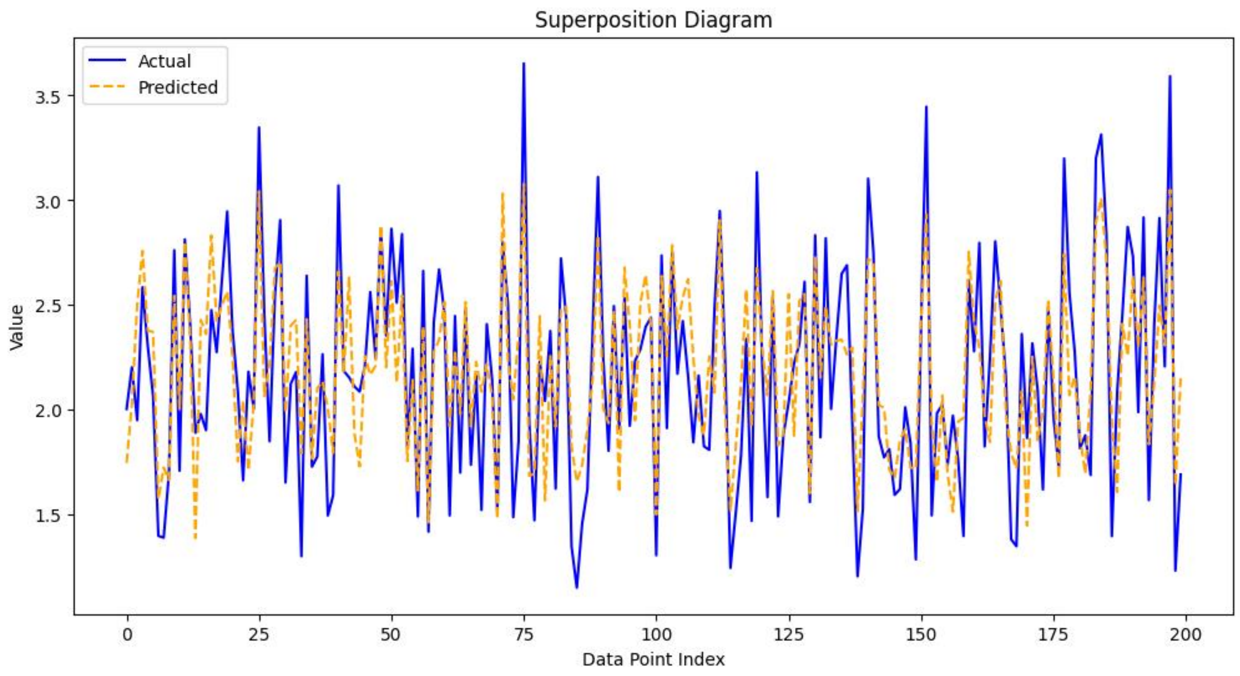
Initial Decrease: At the beginning of training (epochs 0–10), both the training and testing MAE drop significantly. This indicates that the model is learning effectively and improving its performance.

Convergence: After approximately 20 epochs, the MAE for both training and testing stabilizes around 0.25. This suggests that the model has converged and is no longer improving significantly with additional epochs.

Best Epoch: The best model performance (minimum testing MAE) occurs at epoch 97, marked by the red dot. This indicates the point where the model achieves optimal performance on the testing data.

Consistency: The close alignment of training and testing MAE curves throughout the training process indicates that the model generalizes well to unseen data, with no significant overfitting or underfitting observed.

The low and stable MAE values for both training and testing phases, along with the identification of the best epoch, demonstrate the robustness of the model. This graph effectively validates the model's training process and its ability to generalize to new data.



The figure above shows the comparison between the actual values (solid blue line) and the predicted values (dashed orange line) across 200 data points. Key observations from the diagram are as follows:

General Trend Matching: The predicted values closely follow the actual values, indicating that the model successfully captures the underlying patterns in the data.

Alignment: While the two curves generally align, slight deviations can be observed at certain data points, representing prediction errors. However, these deviations are relatively small, suggesting the model has good accuracy.

Fluctuations: Both the actual and predicted values exhibit similar levels of fluctuation, demonstrating the model's ability to adapt to varying data characteristics.

Insights on Noise: The alignment between the two lines suggests the model is robust against noise in the data, maintaining a strong predictive capability.

The superposition of actual and predicted values demonstrates the effectiveness of the model in accurately capturing and predicting trends. While minor discrepancies exist, the overall agreement highlights the model's reliability and precision.

5. Key Observations:

1. **Training Performance:**

   The training MAE consistently decreased with more epochs, signifying effective learning by the model.

   The model successfully minimized the error in the training data without significant overfitting.

2. **Testing Performance:**

   - The testing MAE fluctuated across epochs, with a clear point (optimal epoch) where the testing error was minimized.
   - Beyond the optimal epoch, overfitting became evident, as testing MAE started increasing while training MAE continued decreasing.

3. **Superposition Analysis:**

   - The superposition diagram reveals that the predicted outputs closely follow the actual outputs, particularly around the optimal epoch.
   - Minor deviations are observed, but the overall alignment reflects a high degree of accuracy.

6. Conclusion:

The neural network model demonstrated strong performance in both training and testing phases, achieving a testing MAE of **0.2546** at the optimal epoch (97). The training MAE consistently decreased, indicating effective learning, while the testing MAE stabilized, showcasing good generalization.

The MAE curves highlighted the balance between underfitting and overfitting, with no significant divergence between training and testing errors. The superposition diagram further validated the model's predictive accuracy, as predicted values closely matched actual values, with only minor deviations.

This project demonstrates the importance of identifying the optimal epoch to achieve robust predictions while maintaining a balance between training and testing performance. The model effectively captured patterns in the dataset and provided reliable results, supporting its suitability for regression tasks.

7. Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_absolute_error

from google.colab import drive
drive.mount('/content/drive')

df = pd.read_excel('/content/drive/MyDrive/AI//Hw51/51.xlsx')
data = df.values
df

X, y = data[:, :-1], data[:, -1]

X_train, y_train = X[:300], y[:300]
X_test, y_test = X[300:], y[300:]

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = Sequential([
    Dense(64, activation='relu', input_dim=X_train.shape[1]),
    Dense(32, activation='relu'),
    Dense(1)
])er='adam', loss='mse', metrics=['mae'])

best_epoch = 0
min_mae = float('inf')
mae_train_history = []
mae_test_history = []
epochs_range = range(1, 101)

for epoch in epochs_range:
  history = model.fit(X_train, y_train,
validation_data=(X_test,y_test), epochs=1, batch_size=32, verbose=0)

  y_pred_train = model.predict(X_train)
  y_pred_test = model.predict(X_test)
```

```
    mae_train = mean_absolute_error(y_train, y_pred_train)
    mae_test = mean_absolute_error(y_test, y_pred_test)

    mae_train_history.append(mae_train)
    mae_test_history.append(mae_test)

    if mae_test < min_mae:
      min_mae = mae_test
      best_epoch = epoch

print(f"Best epoch: {best_epoch}")
print(f"Test MAE: {test_mae:.4f}")

plt.figure(figsize=(10, 5))
plt.plot(epochs_range, mae_train_history, label='Training MAE')
plt.plot(epochs_range, mae_test_history, label='Testing MAE')

plt.plot(best_epoch, min_mae, 'ro', markersize=8, label=f'Best Epoch
({best_epoch})')

plt.title('MAE Curves')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()

y_pred = model.predict(X_test)
plt.figure(figsize=(12, 6))
plt.plot(y_test, label='Actual', linestyle='solid', color='blue')
plt.plot(y_pred, label='Predicted',  linestyle='dashed',
color='orange')
plt.title('Superposition Diagram')
plt.xlabel('Data Point Index')
plt.ylabel('Value')
plt.legend()
plt.show()
```