Assignment1. Predicting Relative CPU Performance

Jenna Yang (jy3342)
Computer Science, Columbia University
je.yang@cs.columbia.edu

I. Introduction

The increasing complexity of CPUs requires advanced methods to predict relative performance efficiently. This project explores the application of neural networks to forecast CPU performance based on hardware attributes.

II. Data Preprocessing

The dataset [1] contains 209 CPU models with 8 relevant features and 1 target variable, Published relative performance (PRP) . The dataset includes a categorical feature as a vendor name format. The numerical features include machine cycle time, memory size, cache size, and the number of channels, all of which were standardized using StandardScaler to ensure uniformity in data distribution and prevent features with larger numerical ranges from dominating the model training process.

We split the dataset into 80 percent for training and 20 percent for testing to ensure a balance between learning and generalization [2]. The training set provided sufficient data for the model to recognize patterns in CPU attributes, while the test set allowed for a reliable assessment of model performance. An 80:20 split is a standard approach in regression tasks, preventing overfitting while maintaining enough unseen data for evaluation.

III. Model Architecture

The neural network model consists of an input layer that processes ten features after preprocessing. Baseline architecture has two hidden layers, where the first hidden layer consists of 64 neurons with a ReLU activation function, followed by a dropout layer with a rate of 0.2 to prevent overfitting. The second hidden layer has 64 neurons with a ReLU activation function, followed by another dropout layer with a rate of 0.2. The output layer consists of a single neuron with a linear activation function, targeting for regression tasks.

The model was trained using the Adam optimizer with a learning rate of 0.001. The loss function used was Mean Squared Error (MSE) to measure the deviation between the predicted and actual CPU performance values. The model was trained on NVIDIA RTX 4090 with a batch size of 16 and ran for 100 epochs to ensure convergence and stability during training.

IV. Performance Metric

The performance of the model was evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), and the R-squared (R2) metric. The MAE provides an intuitive measure of the average absolute difference between predicted and actual values, while the MSE penalizes larger errors more heavily, making it suitable for evaluating regression models. The R-squared score measures how well the model explains the variance in the target variable, with a higher score indicating better predictive performance.

TABLE I
Performance metric with varying feature sizes, number of layers, and leraning rate.

	MAE	MSE	R2
64 units	33.88	4923.72	0.90
128 units	31.66	4254.07	0.92
256 untis	32.20	4151.57	0.92
1 layer	49.32	11335.28	0.78
2 layers	35.33	5058.70	0.90
3 layers	31.95	4167.66	0.92
Lr 0.1	34.18	4865.38	0.9
Lr 0.01	28.44	3450.59	0.93
Lr 0.001	33.31	4771.06	0.91
Lr 0.0005	38.04	5539.07	0.89

V. Discussion

The model's performance was explored by varying the number of features, the number of layers, and the learning rate. The performance metrics of MAE, MSE, and R2 for all the scenarios are summarized in Table I. When increasing the number of units per layer from 64 to 128, the MAE and MSE decreased, and the R2 score improved slightly, indicating better predictive performance. However, further increasing the number of units to 256 did not yield significant improvements, suggesting that a model complexity beyond a certain threshold may not always lead to better generalization.

The impact of the number of layers was also evaluated. A single-layer network resulted in significantly higher errors with the lowest R2 score, demonstrating that a deeper architecture was necessary for capturing the relationships within the dataset. Adding a second layer significantly reduced errors, and adding a third layer improved performance slightly.

The learning rate was another key hyperparameter explored. A very high learning rate of 0.1 led to suboptimal results, as the model struggled to converge. A learning rate of 0.01 produced the best results, achieving the lowest MAE and MSE, while a lower learning rate of 0.0005 led to worse performance, likely due to slow convergence and ineffective optimization.

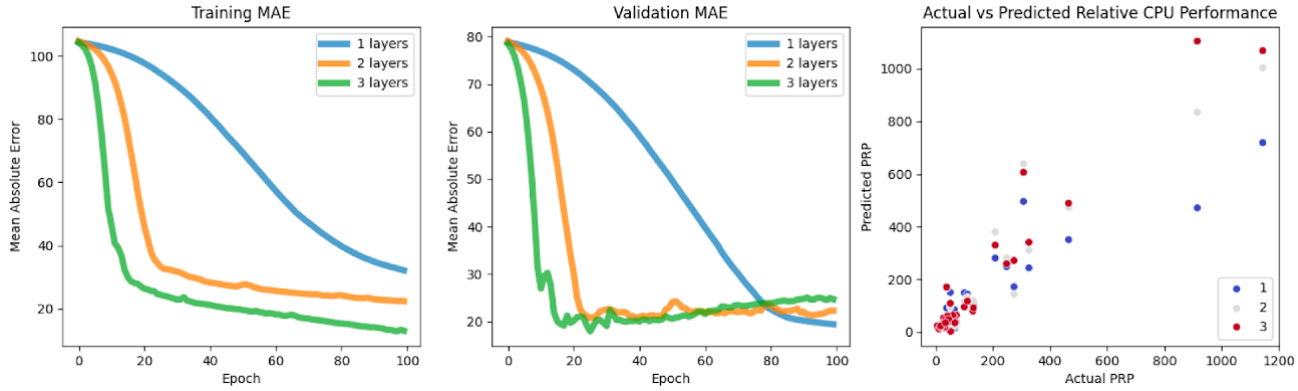

Fig. 1. Training and evaluation loss comparison with varying the number of layers.

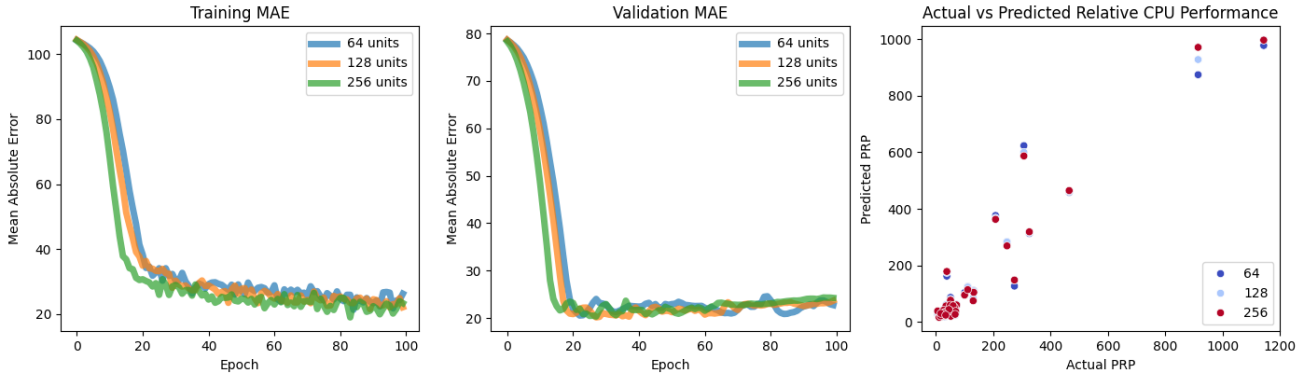

Fig. 2. Training and evaluation loss comparison with varying the feature size of the layer.

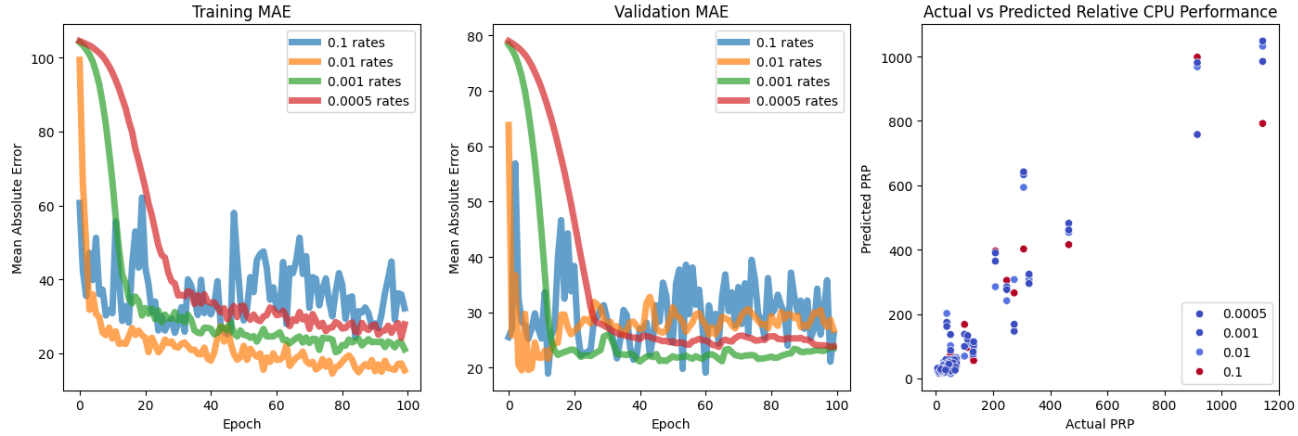

Fig. 3. Training and evaluation loss comparison with varying the learning rates of model.

Figure 1, 2, and 3 show the training and evaluation loss and PRP with varying model design. Training MAE shows the convergence and model stability. The actual-predicted PRP values highlight the model's effectiveness. The predicted values align well with the actual values for lower and mid-range CPUs, but deviations are more noticeable for CPUs with PRP values exceeding 600. These discrepancies indicate the need for either additional features or a more complex model architecture to better capture the variations in high-performance processors.

REFERENCES

[1] J. Feldmesser, "Computer Hardware." UCI Machine Learning Repository, 1987. DOI: https://doi.org/10.24432/C5830D.

[2] Q. H. Nguyen, H.-B. Ly, L. S. Ho, N. Al-Ansari, H. V. Le, V. Q. Tran, I. Prakash, and B. T. Pham, "Influence of data splitting on performance of machine learning models in prediction of shear strength of soil," *Mathematical Problems in Engineering*, vol. 2021, no. 1, p. 4832864, 2021.