Jatilima : Jurnal Multimedia Dan Teknologi Informasi
Volume 07, Number 06, 2026, doi.org/10.54209/jatilima.v7i06.2159
E.SSN : 2721-1800 (Online)
https://journal.cattleyadf.org/index.php/jatilima/index

Round-Trip Time Estimation Using Hybrid Neuro-Fuzzy Based On Subtractive Clustering

Hassan Rizky Putra Sailellah[1], **Suma Danu Ristianto**[2]
[1,2]Telkom University, Indonesia

Article Info
Keywords:
Round-Trip Time,
Quality of Service,
Adaptive Neuro-Fuzzy Inference System,
Estimation,
Subtractive Clustering.

ABSTRACT
Round-trip time (RTT) is a key latency indicator for quality-of-service (QoS) control and task orchestration in cloud–edge systems, yet its strong time variability makes reactive policies brittle. Although machine-learning and neuro-fuzzy predictors have been explored, existing RTT studies often rely on low-dimensional inputs or suffer rule explosion when rich TCP-state features are used, and many reports omit a unified, leakage-free evaluation and practical runtime analysis. This study aims to develop an accurate yet compact RTT predictor by combining an Adaptive Neuro-Fuzzy Inference System (ANFIS) with subtractive-clustering-based initialization, and to quantify the resulting accuracy–complexity–runtime trade-off. A controlled dataset was generated in Mininet using a dumbbell topology with injected delays (1–1000 ms). In total, 100,000 raw RTT measurements (100 per run across 1,000 runs) were collected and aggregated into 1,000 supervised samples paired with TCP-state features. Experiments followed a fixed 60/10/30 train/validation/test split, train-only feature standardization, train-only target normalization with inverse transformation for reporting, and validation-based checkpoint selection. The ANFIS baseline (radius r = 0.5, 19 rules) achieved RMSE/ MAE/ MAPE/ R2 of 1191.63/ 751.02/ 0.001921/ 0.999996 on validation and 1207.23/ 664.70/ 0.001311/ 0.999996 on testing. Training required 546.91 s, while inference remained lightweight (0.0846 s for 100 validation samples and 0.1493 s for 300 test samples). These findings indicate that ANFIS with subtractive clustering can deliver accurate and low-latency RTT prediction suitable for QoS-aware orchestration pipelines when training is performed offline.

Corresponding Author:
Hassan Rizky Putra Sailellah
Telkom University
Bandung, Indonesia
hassanrizkyhrs@telkomuniversity.ac.id

INTRODUCTION

Cloud and edge computing have become the default execution environments for latency-sensitive services, including real-time analytics, interactive applications, and IoT backends (Sailellah et al., 2025). In such systems, network latency directly impacts Quality of Service (QoS), user experience, and the effectiveness of resource orchestration decisions (Sailellah & Nuha, 2024; Zhang et al., 2023). As orchestration policies increasingly operate under strict performance guarantees and resource constraints, they require accurate latency-

related signals to enable proactive control rather than purely reactive adjustment (Larrenie et al., 2022).

A practical and widely used latency indicator is the round-trip time (RTT), defined as the time required for a packet to travel from a sender to a receiver and return with an acknowledgment. However, RTT is highly time-varying due to congestion, routing changes, cross-traffic, queueing dynamics, and dynamic endpoint conditions (Pasco et al., 2023). This variability makes purely reactive strategies that rely only on the latest measurement less reliable, particularly when systems must anticipate imminent degradation to prevent QoS violations and reduce the risk of service instability or downtime (Aldhyani et al., 2020).

Consequently, short-term RTT prediction has attracted increasing attention as a key enabler for proactive QoS control and computation offloading (Putra et al., 2024). Predicting RTT one step (or a few steps) ahead allows schedulers and orchestration layers to anticipate congestion and adapt decision policies more robustly than thresholding on the current observation alone (Khababa et al., 2025). In this direction, (Damaševičius & Sidekerskienė, 2020) proposed a hybrid neuro-fuzzy approach for short-term RTT prediction in cloud environments, motivating the use of neuro-fuzzy modeling to capture non-linear dynamics in RTT behavior.

Neuro-fuzzy models are attractive because they combine the learning capability of neural networks with the rule-based structure of fuzzy inference, offering a balance between expressiveness for nonlinear patterns and interpretability through human-readable fuzzy rules (Cheng et al., 2021). A well-established framework in this family is the Adaptive Neuro-Fuzzy Inference System (ANFIS), which implements a Sugeno-type fuzzy inference system whose parameters can be learned via a hybrid training procedure that alternates between estimating consequent parameters and updating premise parameters (Oladipo et al., 2024).

Despite these advantages, a practical challenge arises when the input dimensionality is high. Constructing an initial fuzzy inference system via grid partitioning can lead to an exponential increase in the number of rules (the curse of dimensionality), making training and deployment impractical. To keep ANFIS feasible in high-dimensional settings, this study initializes the fuzzy inference system using subtractive clustering, which constructs rules from data-driven cluster structures and controls model complexity through a radius (influence range) parameter (Barbierato et al., 2020).

However, prior RTT prediction studies typically do not state clearly (i) how model complexity is controlled when the input feature space is expanded with TCP-state variables, (ii) how sensitive the predictor is to the accuracy–complexity trade-off induced by rule generation, and (iii) whether the resulting predictor is practical for low-latency decision loops when measured with a consistent, leakage-free protocol. Accordingly, this paper addresses the following research question: can ANFIS initialized via subtractive clustering provide accurate short-term RTT prediction from high-dimensional TCP-state features while keeping the rule base compact and inference lightweight?

The objectives of this study are to (1) build a hybrid neuro-fuzzy RTT predictor using ANFIS with subtractive clustering, (2) evaluate it under a unified and reproducible protocol with strict train-only preprocessing, and (3) analyze the radius-driven trade-off between accuracy, number of rules, and training runtime, including comparison with conventional regression baselines.

The main contributions are:

- A reproducible RTT regression dataset and pipeline based on Mininet dumbbell-topology simulations with injected delay and TCP-state features.
- A compact ANFIS baseline initialized with subtractive clustering to mitigate rule explosion in high-dimensional inputs.
- A comprehensive evaluation that reports both accuracy metrics (RMSE, MAE, MAPE, R2) and practicality metrics (training and inference time), plus radius sensitivity analysis.

This paper formulates RTT prediction as a supervised regression problem with high-dimensional input features and evaluates a hybrid neuro-fuzzy baseline implemented as ANFIS with subtractive-clustering-based initialization. The dataset is collected from a controlled network simulation built in Mininet using a dumbbell topology, a standard structure for congestion-control analysis (Ghosh et al., 2025). The simulation is executed for 1000 runs with injected delays; for each run, 100 RTT measurements are recorded as raw records and then aggregated to form one supervised sample paired with TCP-state features, resulting in 1000 regression samples for learning. To ensure fair and reproducible evaluation, the experimental protocol uses a fixed 60/10/30 train/validation/test split with a fixed random seed, standardizes inputs using training statistics only, normalizes the target RTT using training statistics for stable training, and reports performance after inverse transformation on the original RTT scale.

The evaluation is reported using both accuracy and practicality criteria. Specifically, we report RMSE, MAE, MAPE, and R2 on validation and testing sets, and we additionally measure training time and inference time to assess suitability for real-world QoS-aware decision pipelines. Beyond aggregate metrics, we provide diagnostic analyses, including RTT distribution across splits, learning curves, parity plots, residual analyses, and empirical error distributions to support robust interpretation of generalization behavior. We further conduct a radius sensitivity study to characterize the accuracy–complexity–runtime trade-off induced by subtractive clustering.

METHODS

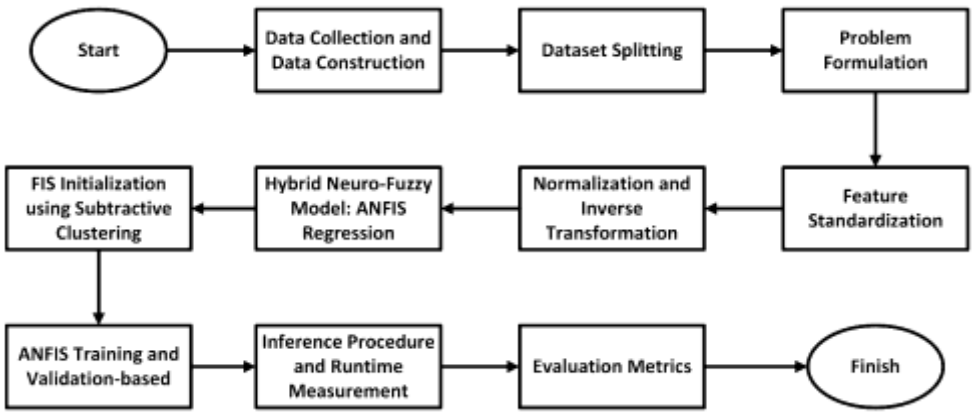


Figure 1. Flowchart Diagram Process

Data collection and dataset construction

A controlled dataset was created using a virtual network topology implemented in Mininet on Ubuntu running inside VirtualBox (Kulkarni et al., 2025). Figure 2 is the emulated network follows a dumbbell topology, four end hosts connected through two main routers and a single shared backbone link commonly adopted for congestion-control and fairness experiments (Chatterjee et al., 2021). During each run, traffic was generated between hosts while propagation delays were injected in the range of 1 ms to 1000 ms. For each run, the system recorded RTT and a set of TCP-state variables including time stamp, source and destination addresses, next_seq, unacknowledged, congestion window (cwnd), source window (snd_wnd), and receive window (rcv_wnd) (Sailellah et al., 2026).

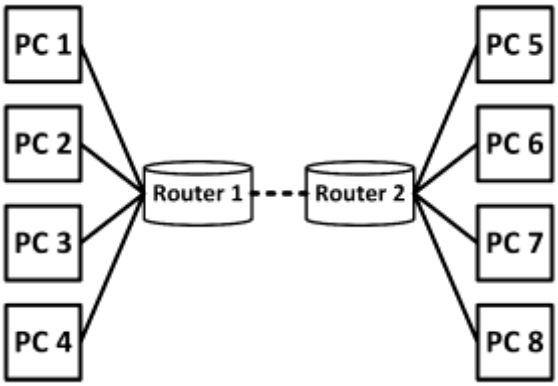


Figure 2. Dumbbell Topology Architecture

The experiment was executed 1000 runs. In each run, 100 RTT measurements (raw records) were collected, producing 100,000 raw RTT records in total. For supervised regression, the learning dataset was formed by aggregating the 100 RTT measurements within each run into a single target value and pairing it with the corresponding TCP-state features. Consequently, the regression pipeline employs 1,000 samples, with each sample representing a single execution, whereas 100,000 denotes the quantity of raw round-trip time (RTT) measurements gathered throughout the data acquisition phase. RTT, which is the duration between packet transmission and its acknowledgment receipt, serves as a crucial metric for assessing network latency and congestion; elevated RTT values generally signify compromised network performance and a diminished quality of experience (Thapliyal, 2024).

Dataset splitting and unified evaluation protocol

The dataset, comprising 1000 samples, was partitioned into training, validation, and testing subsets to facilitate model training, hyperparameter tuning, and an impartial final assessment. In detail, the training set encompassed 600 samples, the validation set 100 samples, and the testing set 300 samples, thereby adhering to a 6:1:3 ratio. Each sample index was exclusively assigned to a single split, with no instances of overlap, thus guaranteeing that no single run contributed records to multiple subsets. Hyperparameters are selected using training/validation only, while the test set is held out and used only once for final reporting.

Let Ndenote the total number of samples ($N = 1000$). Let $\mathcal{D}_{train}$, $\mathcal{D}_{val}$, and $\mathcal{D}_{test}$denote the training, validation, and testing sets with sizes $N_{train} = 600$, $N_{val} = 100$, and $N_{test} = 300$, respectively. This unified protocol is used consistently for all models and all plots reported in the Results and Discussion section.

Problem formulation (RTT regression)

RTT prediction is formulated as supervised regression. Let $\mathbf{x}_i \in \mathbb{R}^d$be the feature vector of the i-th sample and $y_i \in \mathbb{R}$be its corresponding RTT target, where $d = 99$in this study. The goal is to learn a nonlinear function $f(\cdot)$that maps $\mathrm{x}_i$to an estimated RTT $\hat{y}_i$:

$$\hat{y}_i = f(\mathbf{x}_i). \tag{1}$$

The i is the sample index, $i = 1,2, \dots, N$, N is the total number of samples, x_i is the input feature vector for sample i, d is the feature dimension, $y_i$is the true RTT target for sample i, $\hat{y}_i$ is the predicted RTT for sample i, $f(\cdot)$ is the learned regression model

Train-only feature standardization

To prevent data leakage and ensure comparability, features are standardized using statistics computed only on the training set. Let $\mu_X \in \mathbb{R}^d$be the vector of per-feature means and $\sigma_X \in \mathbb{R}^d$be the vector of per-feature standard deviations computed on $\mathcal{D}_{train}$:

$$\boldsymbol{\mu}_X = \frac{1}{N_{train}} \sum_{i \in \mathcal{D}_{train}} \mathbf{x}_i \,, \boldsymbol{\sigma}_X = \sqrt{\frac{1}{N_{train} - 1} \sum_{i \in \mathcal{D}_{train}} (\mathbf{x}_i - \boldsymbol{\mu}_X)^2}. \tag{2}$$

The standardized feature vector $\mathrm{x}_i^{(s)}$is:

$$\mathbf{x}_i^{(s)} = \frac{\mathbf{x}_i - \boldsymbol{\mu}_X}{\boldsymbol{\sigma}_X + \epsilon}. \tag{3}$$

Where μ_X is the vector of training-set means (one mean per feature), σ_X is the vector of training-set standard deviations (one std per feature), N_{train} is the number of training samples (600), $\sum$ is the summation over training indices, $(\cdot)^2$ is the element-wise square for vectors, $\mathrm{x}_i^{(s)}$ is the standardized feature vector of sample i, ϵ is the small constant for numerical stability (set to 10^{-12}), and division is the element-wise because $\sigma_X$is a vector.

Train-only target normalization and inverse transformation

The RTT target is normalized using training-set statistics to stabilize ANFIS training. Let μ_Y and $\sigma_Y$be the mean and standard deviation of targets on $\mathcal{D}_{train}$:

$$\mu_Y = \frac{1}{N_{train}} \sum_{i \in \mathcal{D}_{train}} y_i \,, \sigma_Y = \sqrt{\frac{1}{N_{train} - 1} \sum_{i \in \mathcal{D}_{train}} (y_i - \mu_Y)^2}. \tag{4}$$

The normalized target $y_i^{(n)}$is:

$$y_i^{(n)} = \frac{y_i - \mu_Y}{\sigma_Y + \epsilon}. \tag{5}$$

After ANFIS produces predictions in normalized space $\hat{y}_i^{(n)}$, results are transformed back to the original RTT scale:

$$\hat{y}_i = \hat{y}_i^{(n)} (\sigma_Y + \epsilon) + \mu_Y. \tag{6}$$

Where μ_Y is the mean of RTT targets computed on training data, σ_Y is the standard deviation of RTT targets computed on training data, $y_i^{(n)}$ is the normalized RTT target for sample i, $\hat{y}_i^{(n)}$ is the predicted RTT in normalized scale, $\hat{y}_i$ is the predicted RTT in original (denormalized) scale, and ϵ is the small constant preventing division by zero and improving numerical stability

Hybrid Neuro-Fuzzy model: ANFIS regression

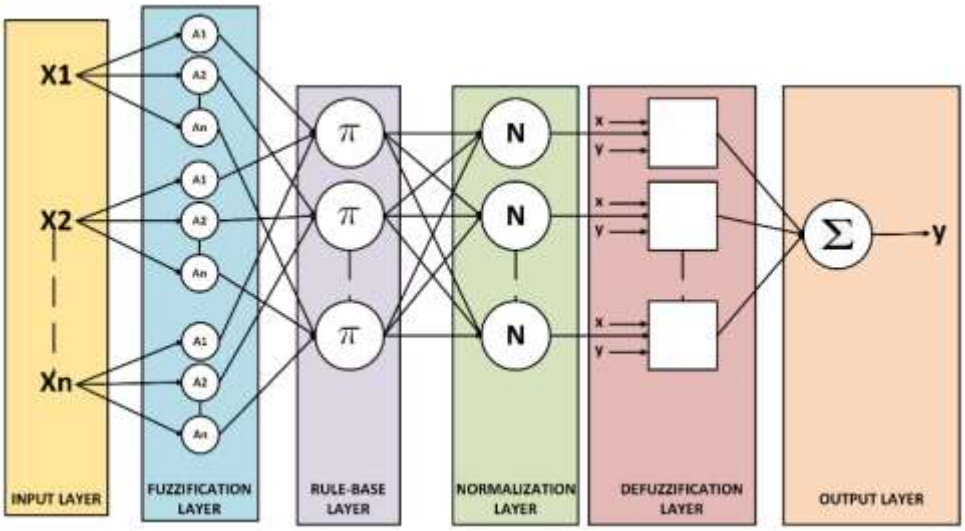


Figure 3. Architecture of the ANFIS (Sugeno-type) predictor

The Hybrid Neuro-Fuzzy baseline is implemented using an Adaptive Neuro-Fuzzy Inference System (ANFIS) can see in Figure 3 with a first-order Sugeno fuzzy inference structure (Damaševi˘cius & Sidekerskiene, 2020). For an ANFIS model with R rules, the normalized output is computed as a weighted average of rule consequents:

$$\hat{y}^{(n)} = \frac{\sum_{r=1}^{R} w_r \, z_r}{\sum_{r=1}^{R} w_r}. \tag{7}$$

Where R is the number of fuzzy rules in the model, r is the rule index, $r = 1,2, \ldots, R.$, w_r is the firing strength (activation weight) of rule r, derived from membership functions in the antecedent part, z_r is the consequent output of rule r(for first-order Sugeno, typically a linear function of inputs), $\hat{y}^{(n)}$ is the ANFIS output in normalized target space, and $\sum_{r=1}^{R}$ is the sum across all rules.

FIS initialization using subtractive clustering (genfis2)

Because grid partitioning becomes infeasible in high-dimensional settings, the initial fuzzy inference system (FIS) is generated using subtractive clustering (Mousavi et al., 2021). The clustering radius $r \in (0,1]$controls the granularity of clustering: smaller $r$tends to produce more clusters and thus more fuzzy rules, while larger $r$yields fewer clusters and a simpler rule base. The initial FIS is:

$$\text{FIS}_0 = \text{genfis2}(\mathbf{X}_{train}^{(s)}, \mathbf{y}_{train}^{(n)}, r). \tag{8}$$

Where FIS_0 is the initial fuzzy inference system before ANFIS training, $\text{X}_{train}^{(s)}$ is the standardized training feature matrix, $\text{y}_{train}^{(n)}$ is the normalized training target vector, r is the

subtractive clustering radius hyperparameter, and genfis2(·) is the MATLAB function that builds a Sugeno-type FIS using subtractive clustering.

ANFIS training and validation-based checkpoint selection

ANFIS is trained for a fixed number of epochs Eusing training data, while the validation set is used as checking data to support early selection of the best model. Let FIS(e)denote the ANFIS model after epoch e. The selected final model $\text{FIS}_{final}$is the checking model that minimizes validation error:

$$\text{FIS}_{final} = \arg\min_{e\in\{1,\dots,E\}} \mathcal{L}_{val}(\text{FIS}(e)), \tag{9}$$

Where $\mathcal{L}_{val}$ is the validation loss measured on $\mathcal{D}_{val}$in normalized space, E is the total number of training epochs (100 in this study), e is the epoch index, $e = 1,2,\dots,E.$, FIS(e) is the ANFIS model after epoch e, FIS_{final} is the selected final ANFIS model (checking FIS), and arg min is the operator returning the argument (epoch) that minimizes the loss

Inference procedure and runtime measurement

Given FIS_{final}, predictions for validation and test sets are obtained via:

$$\hat{\mathbf{y}}_{val}^{(n)} = \text{FIS}_{final}\left(\mathbf{X}_{val}^{(s)}\right), \hat{\mathbf{y}}_{test}^{(n)} = \text{FIS}_{final}\left(\mathbf{X}_{test}^{(s)}\right). \tag{10}$$

Where $\text{X}_{val}^{(s)}$, $\text{X}_{test}^{(s)}$ is the standardized validation and test feature matrices, $\hat{\text{y}}_{val}^{(n)}$, $\hat{\text{y}}_{test}^{(n)}$ is the predicted targets in normalized space for validation and test sets, and $\text{FIS}_{final}(\cdot)$ is the evaluation of the final ANFIS model on input features. Predictions are then inverse transformed to the original RTT scale using the equation in Section 2.5. Runtime measurements are recorded for training, validation inference, and test inference using wall-clock timing, enabling the execution-time.

Evaluation metrics

All metrics are computed on the original RTT scale using true targets $y_i$and denormalized predictions $\hat{y}_i$, for a subset of size M(validation or test).

- **Root Mean Square Error (RMSE):**

$$\text{RMSE} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2}. \tag{11}$$

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{M}\sum_{i=1}^{M} |\, y_i - \hat{y}_i \,|. \tag{12}$$

- **Mean Absolute Percentage Error (MAPE, 0–1 scale):**

$$\text{MAPE} = \frac{1}{M}\sum_{i=1}^{M} \left| \frac{y_i - \hat{y}_i}{\max(|\, y_i \,|, \epsilon)} \right|. \tag{13}$$

- **Coefficient of determination (R^2):**

$$R^2 = 1 - \frac{\sum_{i=1}^{M}(y_i - \hat{y}_i)^2}{\max\left(\sum_{i=1}^{M}(y_i - \bar{y})^2, \epsilon\right)}, \bar{y} = \frac{1}{M}\sum_{i=1}^{M} y_i. \quad (14)$$

Radius sensitivity analysis (accuracy–complexity–runtime)

A sensitivity study is conducted by sweeping the subtractive clustering radius:

$$r \in \{0.3, 0.4, 0.5, 0.6, 0.7\}. \quad (15)$$

Where r is the subtractive clustering radius, for each radius value, the initial FIS is generated via subtractive clustering, trained using the same epoch count, and selected using validation-based checkpointing. Accuracy is assessed using RMSE on validation and testing subsets, model complexity is quantified by the number of fuzzy rules R, and computational cost is measured using training time. This procedure supports the accuracy–complexity–runtime trade-off analysis reported in the Results & Discussion section.

RESULTS AND DISCUSSION

Radius sensitivity analysis (accuracy–complexity–runtime)

Table 1. ANFIS baseline performance and runtime

Model	Type Data	RMSE	MAE	MAPE	R2	Train/ Tune (s)	Val Time (s)	Test Time (s)	#Rules
Hybrid Neuro-Fuzzy (ANFIS)	Validation Data	1,191	751	0.001921	0.999996	546.9095	0.084580	0.149349	19
	Testing Data	1,207	664	0.001311	0.999996				

Table 1 reports the predictive accuracy and computational cost of the Hybrid Neuro-Fuzzy baseline implemented using ANFIS with subtractive clustering initialization ($r = 0.5$) under the unified evaluation pipeline. The model achieves RMSE of 1191.63 (validation) and 1207.23 (testing), with MAE of 751.02 (validation) and 664.70 (testing). In relative terms, MAPE is 0.001921 on validation and 0.001311 on testing (0–1 scale), indicating a low average relative error. The goodness-of-fit remains very high with $R^2 \approx 0.999996$ on both validation and testing, suggesting that the model explains nearly all variance in the target RTT under the evaluated split. In terms of runtime, training requires 546.91 s, while inference is lightweight (0.08458 s for the validation subset and 0.14935 s for the testing subset), which supports near real-time prediction once training has been completed. The selected fuzzy system contains 19 rules, indicating that high accuracy is obtained without an excessively large rule base.

Distribution consistency across splits

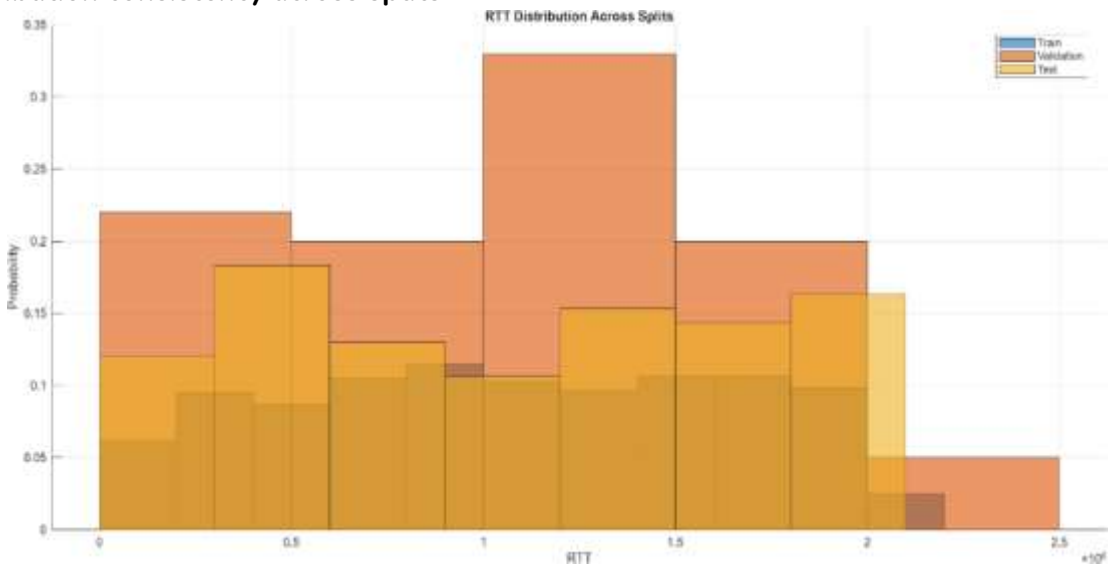


Figure 4. Probability-normalized RTT distributions for training, validation, and testing subsets, illustrating coverage consistency across splits.

Figure 4 shows probability-normalized RTT distributions for the training, validation, and testing subsets. The three distributions overlap substantially, indicating that the fixed split provides comparable RTT coverage across subsets. This overlap reduces the likelihood that the observed test performance is driven by an unrepresentative or systematically easier test set, thereby supporting the validity of the generalization assessment reported in Table 1.

Training dynamics and validation-based model selection

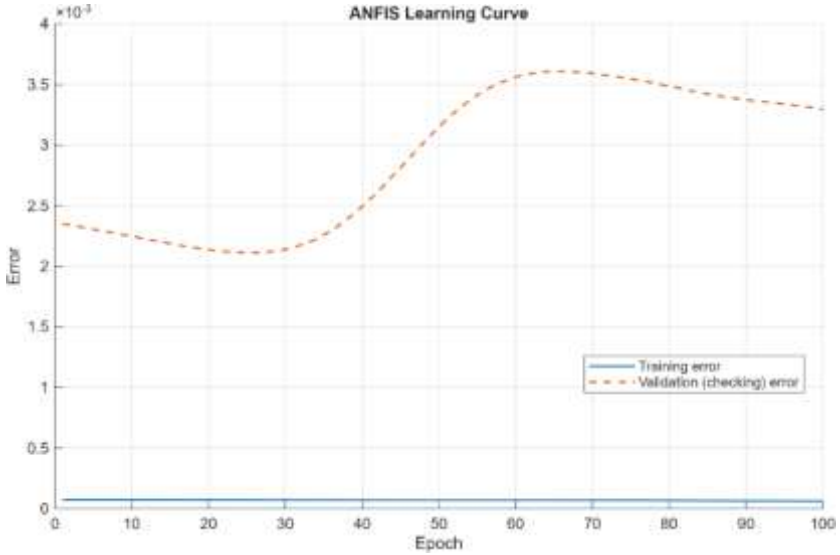


Figure 5. ANFIS learning curve showing training error and validation (checking) error across epochs.

Figure 5 presents the ANFIS learning curve. Training error remains very low, while the validation (checking) error decreases during early epochs and exhibits a gradual increase at

later epochs. This pattern is consistent with a model that can closely fit the training data, which could lead to overfitting if training continues without safeguards. Therefore, the final model is chosen using the checking FIS from the best validation checkpoint, which aligns with the goal of reducing generalization error. The small gap between validation and testing RMSE in Table 1 further indicates stable generalization under this selection strategy.

Prediction fit and calibration

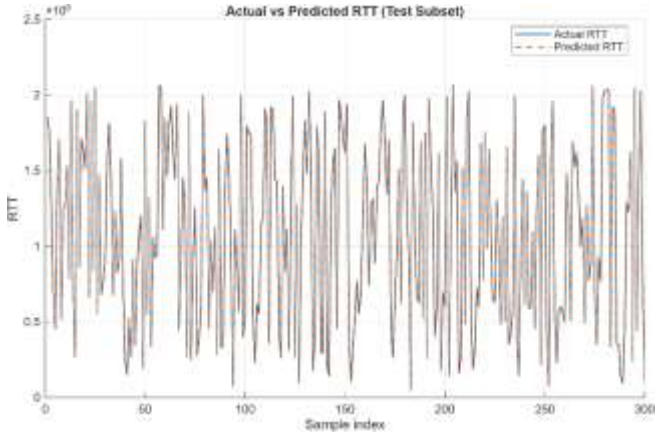


Figure 6. Actual versus predicted RTT on a representative test subset

Figure 6 overlays actual and predicted RTT over a representative subset of the test data and shows close tracking of fluctuations across samples. Figure 7 provides a parity plot for the entire test subset; points lie tightly around the identity line y=x, indicating strong calibration and limited systematic bias. Together, these figures visually support the quantitative performance metrics reported in Table 1, showing that the model captures both the global scale and the local variability of RTT.

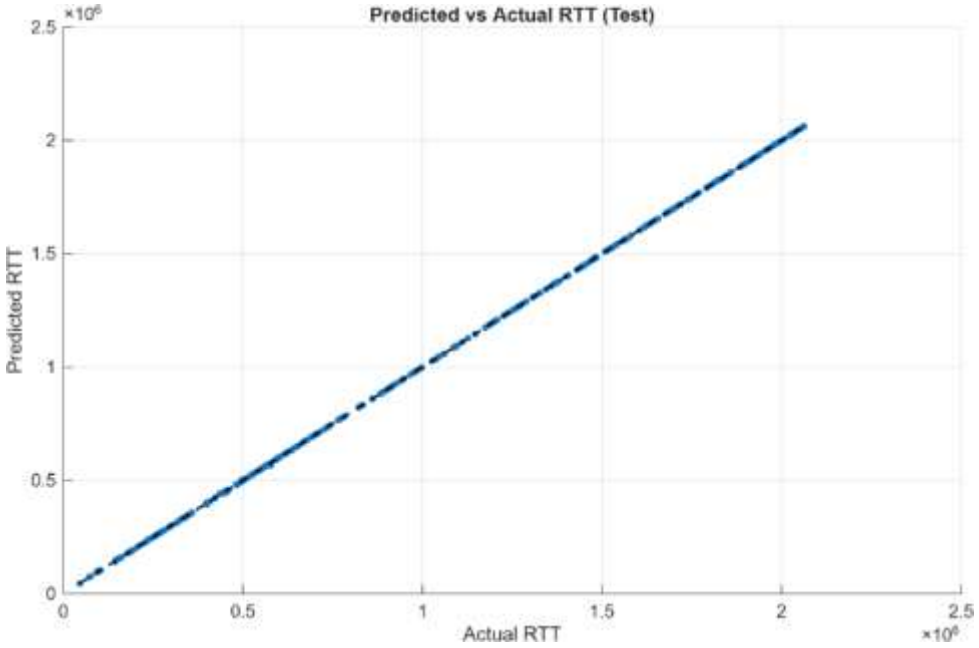


Figure 7. Predicted versus actual RTT on the test set (parity plot)

Error characteristics and reliability analysis

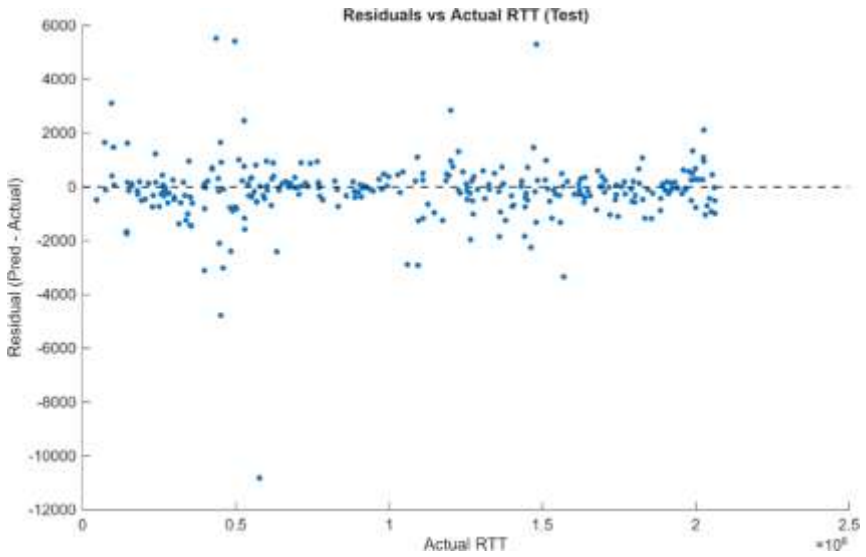
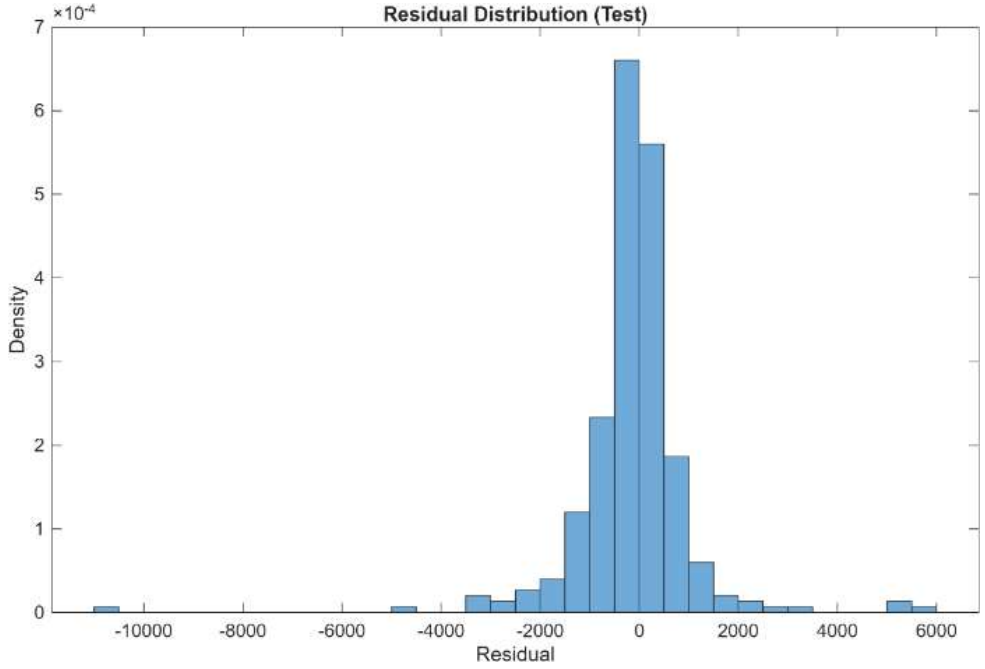
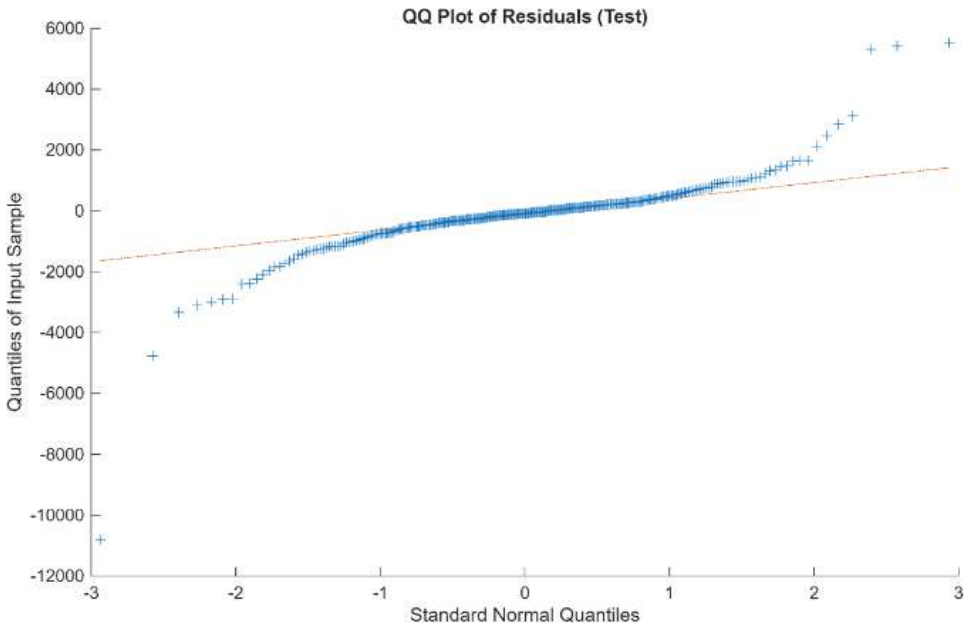


Figure 8. Residuals (Predicted − Actual) versus actual RTT on the test set

Residual diagnostics for the test set are shown in Figures 8–9. Figure 8 plots residuals (Predicted − Actual) versus actual RTT and indicates that residuals are cantered near zero across the RTT range, suggesting limited bias with respect to RTT magnitude. Figure 9(a) shows the residual distribution, where most errors are concentrated near zero with small probability mass in the tails. Figure 9(b) (QQ plot) shows a near-linear trend for central quantiles with deviations at the extremes, implying that residuals are not perfectly Gaussian and that a small number of samples may exhibit larger deviations.



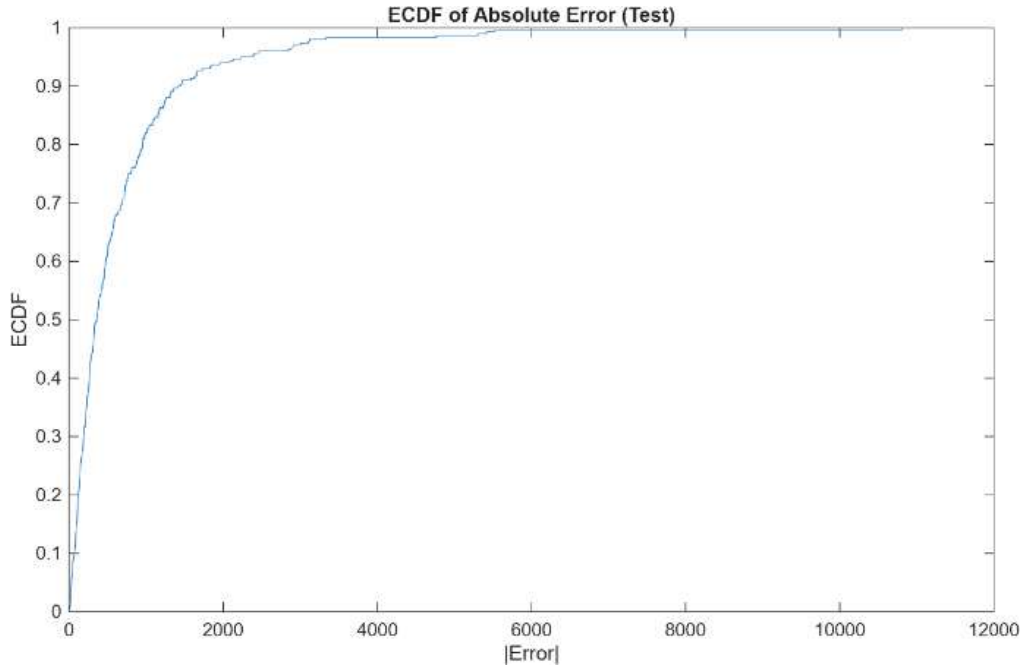
(a)



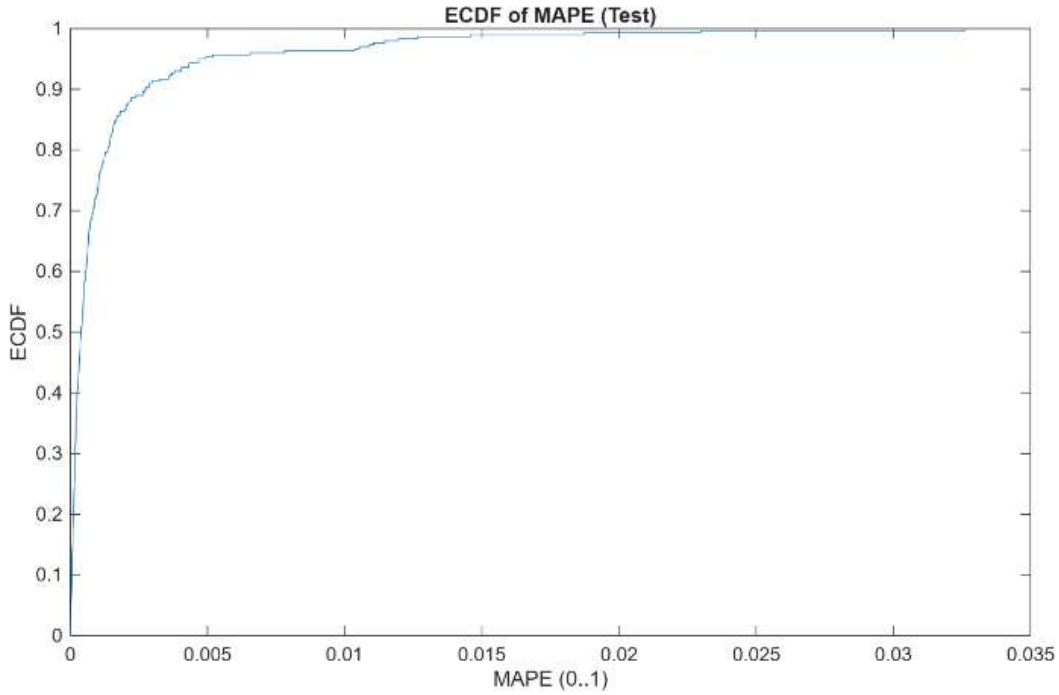
(b)

Figure 9. (a) Residual distribution on the test set (density-normalized histogram) and **(b)** QQ plot of test residuals against standard normal quantiles

To complement point metrics such as RMSE and MAE, Figures 10 report distributional reliability. Figure 10(a) (ECDF of absolute error) indicates that a large portion of samples achieves small absolute errors, while Figure 10(b) (ECDF of MAPE) confirms that relative errors are low for most samples. These distributional views strengthen the interpretation of Table 1 by separating typical performance from rare tail events.



(a)



(b)

Figure 10. (a) ECDF of absolute error |y^-y| on the test set and **(b)** ECDF of per-sample MAPE (0–1 scale) on the test set

Sensitivity of subtractive clustering radius

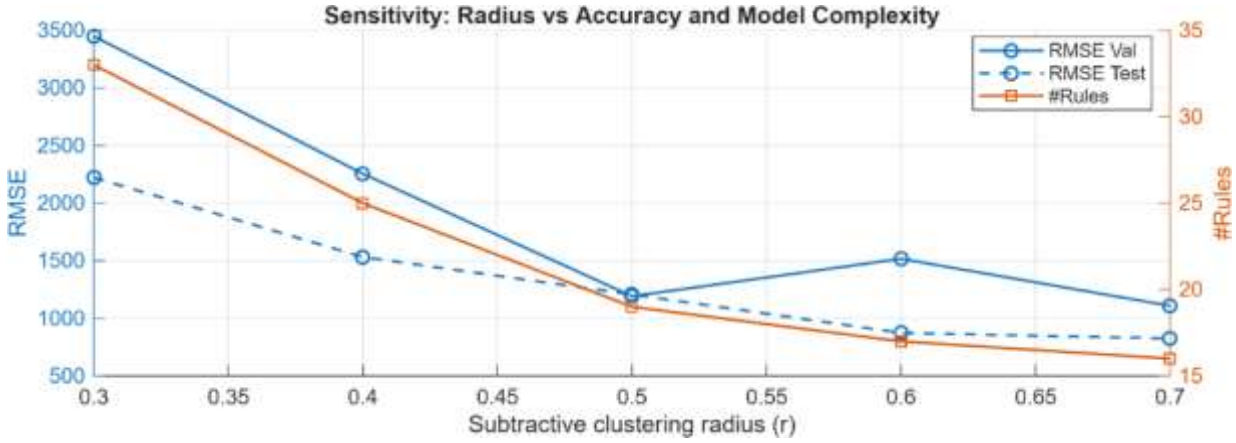


Figure 11. Probability-normalized RTT distributions for training, validation, and testing subsets, illustrating coverage consistency across splits.

Figures 11-12 analyze the effect of the subtractive clustering radius on accuracy, complexity, and training time. Figure 11 shows that decreasing the radius increases the number of fuzzy rules and can affect RMSE on both validation and testing sets. Across the sweep, the number of rules decreases from 33 (at r=0.3) to 16 (at r=0.7), reflecting reduced model complexity with larger radii. Figure 12 shows that training time decreases substantially as the radius increases, consistent with the reduction in rule count and the corresponding reduction in parameters to be learned. These figures indicate that the radius is a key control parameter that determines the accuracy–complexity–runtime trade-off. In this

study, r=0.5is used as a fixed baseline to ensure consistent reporting under the unified pipeline.

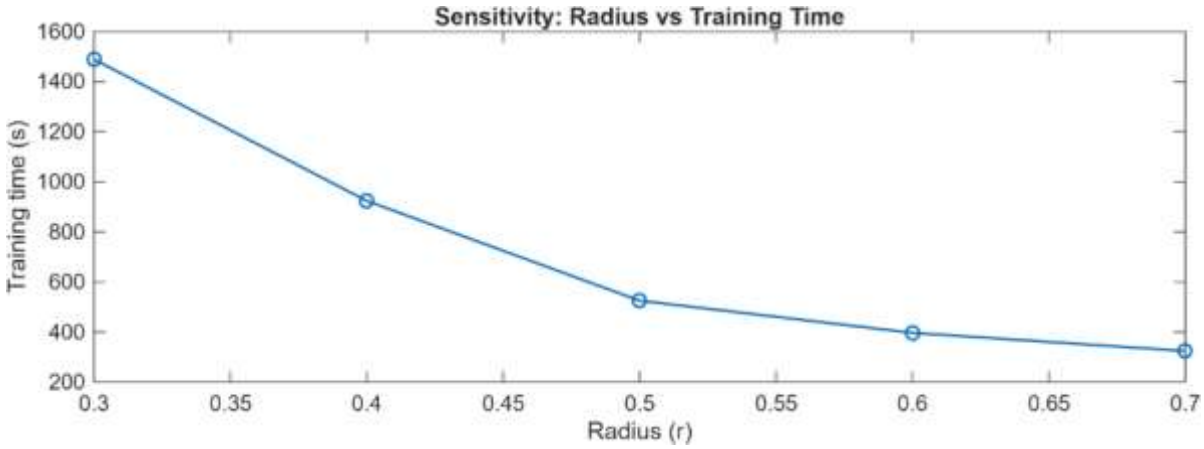


Figure 12. Sensitivity analysis: subtractive clustering radius versus training time

Comparison against conventional regression baselines

Table 2. Conventional regression baselines

Model	Type Data	RMSE	MAE	MAPE	R2	Train/ Tune (s)	Val Time (s)	Test Time (s)
GAM (fitrgam)	Validation Data	31,599	17,675	0.0316960	0.99695	272.3918	0.089907	0.091772
	Testing Data	18,281	12,170	0.0209880	0.99909			
Decision Tree	Validation Data	15,703	10,862	0.0231320	0.99925	1.4642	0.002530 8	0.000583 4
	Testing Data	13,447	9,129	0.0157440	0.99951			
KNN	Validation Data	15,975	10,475	0.0140710	0.99922	9.4485	0.011448	0.037323
	Testing Data	12,164	9,081	0.0142630	0.99960			
Random Forest	Validation Data	10,682	7,692	0.0135390	0.99965	47.2547	0.014903	0.019242
	Testing Data	9,199	6,789	0.0122830	0.99977			
GDBT (LSBoost)	Validation Data	6,038	4,364	0.0077504	0.999888	123.5411	0.035009	0.040067
	Testing Data	6,140	4,418	0.0081372	0.999897			
Hybrid Neuro-Fuzzy (ANFIS)	Validation Data	1,191	751	0.001921	0.999996	546.9095	0.084580	0.149349
	Testing Data	1,207	664	0.001311	0.999996			

Table 2 reports conventional regression baselines (SVR-RBF, GAM, Decision Tree, KNN, Random Forest, and GBDT) under the same unified evaluation protocol. Ensemble techniques, specifically Random Forest and GBDT, demonstrate the highest accuracy among the established baselines, whereas SVR-RBF exhibits the least favorable performance. In contrast to these traditional baselines, the ANFIS baseline, as presented in Table 1, attains significantly reduced RMSE and MAPE values within the assessed split. This suggests that the neuro-fuzzy architecture is adept at modeling the nonlinear associations present within the dataset. This gain in accuracy is accompanied by higher offline training time (Table 1), whereas inference remains lightweight, making the approach suitable for scenarios where training can be performed offline and prediction is required at low latency.

Discussion

The results show that ANFIS initialized with subtractive clustering can model the nonlinear relationship between TCP-state features and RTT with a compact rule base (19 rules at r = 0.5) while maintaining low inference latency. This supports the motivation in the Introduction that neuro-fuzzy models can capture nonlinear latency dynamics without sacrificing interpretability.

From a practicality perspective, the training time (546.91 s) is higher than most conventional baselines, but the inference time is small (sub-second for hundreds of samples), which is consistent with deployment scenarios where model training is performed offline and the predictor is embedded into an online QoS-aware controller or scheduler. Therefore, the proposed baseline is more suitable for offline-trained, online-inference pipelines rather than continuously retrained settings.

The radius sensitivity analysis further clarifies how subtractive clustering mitigates rule explosion: smaller radii generate more clusters and rules, improving representational capacity but increasing training cost, whereas larger radii reduce the rule base and training time at the risk of underfitting. This provides an explicit knob to balance accuracy and computational budget when adapting the predictor to different platforms.

Compared with conventional regression baselines, tree ensembles (Random Forest and GBDT) provide strong performance but still exhibit larger RMSE than the ANFIS baseline under the evaluated split. This suggests that the hybrid neuro-fuzzy structure is effective for the studied feature–target relationship; however, the advantage should be revalidated under broader network dynamics and potential distribution shift.

Limitations of this study include the use of a controlled Mininet environment and the aggregation of 100 raw RTT measurements into one supervised target per run, which may reduce noise and yield optimistic generalization metrics relative to real-world traces. Future evaluation should include additional topologies, cross-traffic patterns, and real measurements, as well as multi-step forecasting and online adaptation to handle non-stationary latency.

CONCLUSION

This paper studied short-term RTT prediction for latency-sensitive edge–cloud services using a hybrid neuro-fuzzy baseline implemented as an Adaptive Neuro-Fuzzy Inference System

(ANFIS) with subtractive-clustering initialization. The dataset was generated from Mininet dumbbell-topology simulations, producing 1000 supervised samples aggregated from 100,000 raw RTT records. A unified and reproducible evaluation pipeline was applied, including a fixed 60/10/30 train/validation/test split (rng = 82), train-only feature standardization, train-only target normalization with inverse transformation for reporting, and validation-based checkpoint selection.

Under this protocol, the ANFIS baseline ($r = 0.5$, 19 rules) achieved RMSE = 1191.63, MAE = 751.02, MAPE = 0.001921, and $R^2 = 0.999996$on the validation set, and RMSE = 1207.23, MAE = 664.70, MAPE = 0.001311, and $R^2 = 0.999996$on the test set. ANFIS training took 546.91 seconds, but inference was quick, taking 0.08458 seconds for validation (100 samples) and 0.14935 seconds for testing (300 samples). This suggests it can be used for low-latency predictions after training. Diagnostic analyses, including learning curves, parity plots, residual inspection, and ECDF-based error distributions, showed that predicted and observed RTT values were very similar. The residuals were mostly near zero, with only small differences. Also, the radius sensitivity study showed that subtractive clustering offers a clear trade-off between accuracy, the complexity of the rule base, and training time.

Future work will extend evaluation to broader network conditions and topologies, investigate multi-step RTT forecasting, and explore online/adaptive updating to address non-stationary latency patterns, alongside comparisons with additional modern learning baselines under distribution shift.

ACKNOWLEDGEMENT

This research is funded by Telkom University under the Grant “SISTEM BUDIDAYA HIDROPONIK MENGGUNAKAN TEKNOLOGI IOT DENGAN FITUR VEGETABLE SELECTOR DAN APLIKASI MOBILE” NO. 701/LIT06/PPM-LIT/2025.

REFERENCE

Aldhyani, T. H., Alrasheedi, M., Alqarni, A. A., Alzahrani, M. Y., & Bamhdi, A. M. (2020). Intelligent hybrid model to enhance time series models for predicting network traffic. IEEE Access, 8, 130431-130451.

Putra, Y. A., Nuha, H. H., & Sailellah, H. R. P. (2024, November). Improving Online Computation Offloading in Mobile-Edge Computing Networks using Deep Reinforcement Learning. In 2024 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT) (pp. 847-852). IEEE.

Barbierato, L., Estebsari, A., Bottaccioli, L., Macii, E., & Patti, E. (2020). A distributed multimodel cosimulation platform to assess general purpose services in smart grids. IEEE Transactions on Industry Applications, 56(5), 5613-5624.

Chatterjee, T., Karmakar, R., Mitra, A., Ghosh, A., & Bhattacharya, S. (2021, July). ICon: How Can We Intelligently Predict TCP Congestion?. In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.

Chen, D., Cai, J., Huang, Y., & Lv, Y. (2021). Deep neural fuzzy system oriented toward high-dimensional data and interpretable artificial intelligence. Applied Sciences, 11(16), 7766.

Damaševičius, R., & Sidekerskienė, T. (2020). Short time prediction of cloud server round-trip time using a hybrid neuro-fuzzy network. Journal of artificial intelligence and systems., 2(1), 133-148.

Ghosh, C. R., Ahsan, N., Islam, M., Noor, R., & MAshrafi, A. (2025). An SDN-based approach using RYU controller for load balancing and performance evaluation in hybrid networks with machine learning algorithms (Doctoral dissertation, Brac University).

Khababa, G., Bessou, S., Seghir, F., Harun, N. H., Almazyad, A. S., Jangir, P., & Mohamed, A. W. (2025). Collaborative filtering techniques for predicting web service qos values in static and dynamic environments: A systematic and thorough analysis. IEEE Access.

Larrenie, P., Bercher, J. F., Lahsen-Cherif, I., & Venard, O. (2022, November). Low Complexity Adaptive ML Approaches for End-to-End Latency Prediction. In International Conference on Machine Learning for Networking (pp. 72-87). Cham: Springer Nature Switzerland.

Mousavi, A., Jalali, M., & Yaghoubi, M. (2021). Adaptive Neuro Fuzzy Networks based on Quantum Subtractive Clustering. arXiv preprint arXiv:2102.00820.

Oladipo, S., Sun, Y., & Amole, A. O. (2024). Investigating the influence of clustering techniques and parameters on a hybrid PSO-driven ANFIS model for electricity prediction. Discover Applied Sciences, 6(5), 265.

Pasco, C. D. R., Bernardini, F., & Antônio, A. D. A. (2023, June). In-network Latency Nowcast Using Data Stream Learning Models. In 2023 International Wireless Communications and Mobile Computing (IWCMC) (pp. 1214-1219). IEEE.

Sailellah, H., & Nuha, H. H. (2024, August). Round-trip time estimation using regularized extreme learning machine. In 2024 International Conference on Artificial Intelligence, Blockchain, Cloud Computing, and Data Analytics (ICoABCD) (pp. 79-84). IEEE.

Sailellah, H. R. P., Nuha, H. H., & Putrada, A. G. (2026). Round-Trip Time Estimation Using Enhanced Regularized Extreme Learning Machine. Network, 6(1), 10.

Sailellah, H. R. P., Santoso, G., & Setyorini (2025, October). Estimating Round-Trip Time Using Machine Learning Models: A Comparative Study of SVM, GAM, and Linear Regression. In 2025 IEEE 9th International Conference on Software Engineering & Computer Systems (ICSECS) (pp. 553-558). IEEE.

Thapliyal, S. (2024, March). Predicting Network Latency Using Adaptive Network Based Fuzzy Inference System. In 2024 3rd International Conference for Innovation in Technology (INOCON) (pp. 1-6). IEEE.

Zhang, W., Feng, M., & Krunz, M. (2023). Latency estimation and computational task offloading in vehicular mobile edge computing applications. IEEE Transactions on Vehicular Technology, 73(4), 5808-5823.