

Applied Computational Intelligence Project Report

David Gao Xia n^o 99907 , João Barreiros C. Rodrigues n^o 99668

Oct. 2024

Fuzzy System

General Architecture

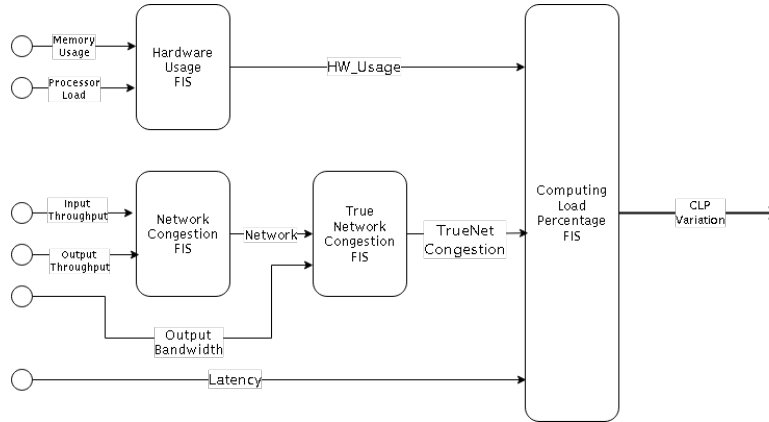


Figure 1: Fuzzy System Architecture

From the 12 given variables we opted to only kept the 6 non-variations, since a Fuzzy System with 6 inputs will be smaller (and therefore simpler to fine-tune and write rules for) and the variations give no absolute current information which is the basis for our decision system.

Similar to the idea of Comb's method, we broke our FIS (Fuzzy Inference System) into smaller, 2 to 3 input FIS. This allowed to have some granularity in their input and output membership sets without the risk of exploding. Our Architecture has 4 FIS.

Hardware Usage FIS

This system aggregates the hardware-related inputs: **Memory Usage** and **Processor Load**. Our rules set is fairly simple, having a 4 linguistic term granularity for inputs and output.

Memory		Processor		Load
Low	Low	Average	High	Critical
Average	Low	Low	Low	Critical
High	Low	Balanced	Balanced	Critical
Critical	Low	Balanced	High	Critical
	Critical	Critical	Critical	Critical

And so are the membership function for this FIS, following the Project guidelines and common sense.

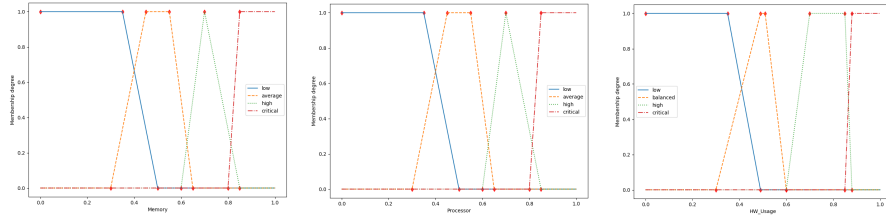


Figure 2: Membership Functions for Hardware Input and Output variables

Network Congestion FIS

This system outputs a “quantitative” network congestion value, based on the I/O throughput of a network.

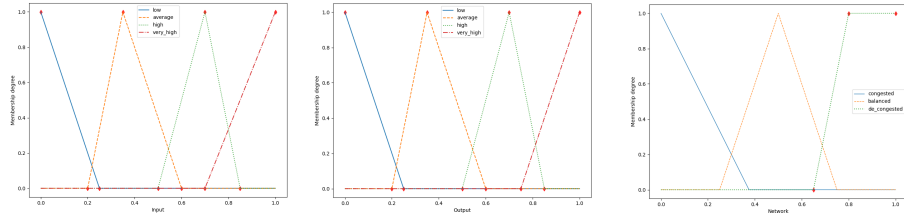
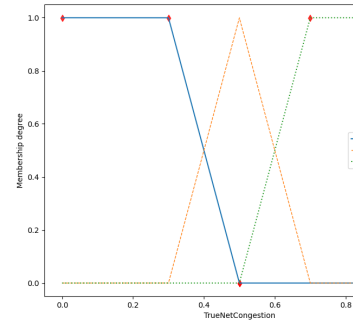
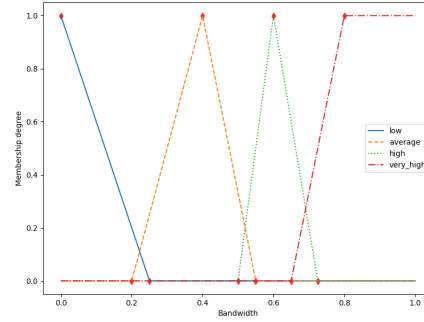


Figure 3: Membership Function for Net. Congestion Input and Output variables

Input Th.		Output		Th.
Low		Average	High	Very High
Low	Decongested	Decongested	Decongested	Decongested
Average	Congested	Balanced	Decongested	Decongested
High	Congested	Congested	High	Decongested
Very High	Congested	Congested	Balanced	Balanced

True Network Congestion FIS

Different from the last FIS the output of this System is not as much a “quantita-



tive” variable as it is and action indicative

Out. Bandwidth	Previous Calculated Net.		
Low	Congested	Balanced	DeCongested
Average	F. Congested	F. Congested	F. Congested
High	F. Congested	Balanced	Balanced
Very High	F. Congested	Balanced	Balanced
	DeCongest.	DeCongest.	Balanced

Final CLP FIS

Since this System takes 3 inputs we reduced the Latency granularity to 3 linguistic terms, so to simplify the rule set. The High Latency Membership starts growing faster with a latency of 0.8 ms, since this is significative of a very poor network connection.

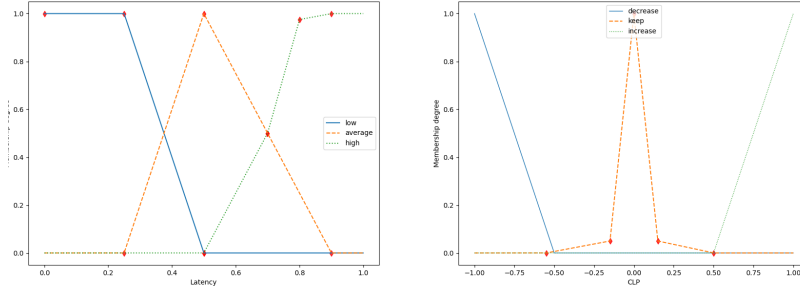


Figure 4: Membership Function for CLP-exclusive Input and Output variables

The rule set for the final FIS is slightly more complex. By fixing the Latency variable we can obtain the rule set tables. For a **Low Latency** value:

HW Usage		True Net. Congestion	
	F. Congested	Balanced	DeCongest.
Low	Increase	Increase	Increase
Balanced	Increase	Keep/Increase	Keep
High	Increase	Decrease	Decrease
Critical	Decrease	Decrease	Decrease

For a **Average Latency** value:

HW Usage		True Net. Congestion	
	F. Congested	Balanced	DeCongest.
Low	Increase	Increase	Increase
Balanced	Increase	Keep/Increase	Keep
High	Increase	Keep	Decrease
Critical	Decrease	Decrease	Decrease

For a **High Latency** value:

HW Usage		True Net. Congestion	
	F. Congested	Balanced	DeCongest.
Low	Increase	Increase	Increase
Balanced	Increase	Increase	Increase
High	Increase	Increase	Increase
Critical	Decrease	Decrease	Decrease

The primary rule is the hardware protection-rule, since avoiding malfunctions/crashing on the Edge device and security of the network are our priorities - If the Hardware has reached criticality the CLP must be decreased.

Another rule obviously-prioritized is that when the Hardware Usage is Low or the Network is Fully congested (i.e. very difficult to de-congest) the CLP must be increased(except of course when conflicting with the first rule)

Another clear rule is that when latency is considered too high, priority is given to local data processing (which should translate into a CLP increase), once again exception is given when conflicting with rule number one.

Performance Metrics

Our fuzzy system solution has a MSE (Mean Squared Error) of **0.0058** , which indicates a seemingly strong performance for the provided dataset. The comparison between the computed Computing Load Percentage variation (CLPv) and the given reference can be observed below.

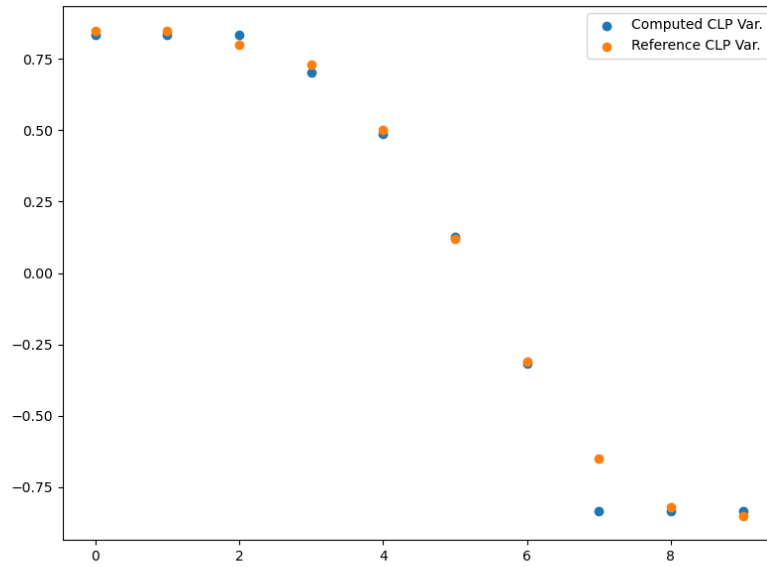


Figure 5: Computed CLPv compared to reference CLPv using initial dataset

Neural Network

To build our neural network model, we started by generating a random dataset with 10,000 rows. This dataset was processed using the Fuzzy System, which provided the input features for the neural network.

The dataset was then split into three subsets: 70% of the data was allocated for training, 15% for validation, and the remaining 15% for testing. The training set was used to train the neural network through cross-validation, ensuring that the model generalized well across different splits of the data. After completing cross-validation, we used the validation set to fine-tune and identify the optimal hyperparameters for the neural network. Finally, we evaluated the model's performance on the test set to assess its final predictive accuracy.

Architecture

The final neural network model consisted of six input features and two hidden layers. The first hidden layer contained 18 neurons, while the second hidden layer consisted of 5 neurons. We selected the logistic function as the activation function. The solver used for optimizing the network was the lbfgs algorithm.

This configuration of the neural network was chosen based on the best performance observed during the validation phase, ensuring that the final model achieved a strong balance between accuracy and generalization. We then tested this model on the test set to evaluate its real-world performance.

Performance Metrics

Conclusions