Master's in Electrical and Computer Engineering

Applied Computational Intelligence

First Project Report

David Gao Xia nº 99907 , João Barreiros C. Rodrigues nº 99668

Fuzzy System

Architecture

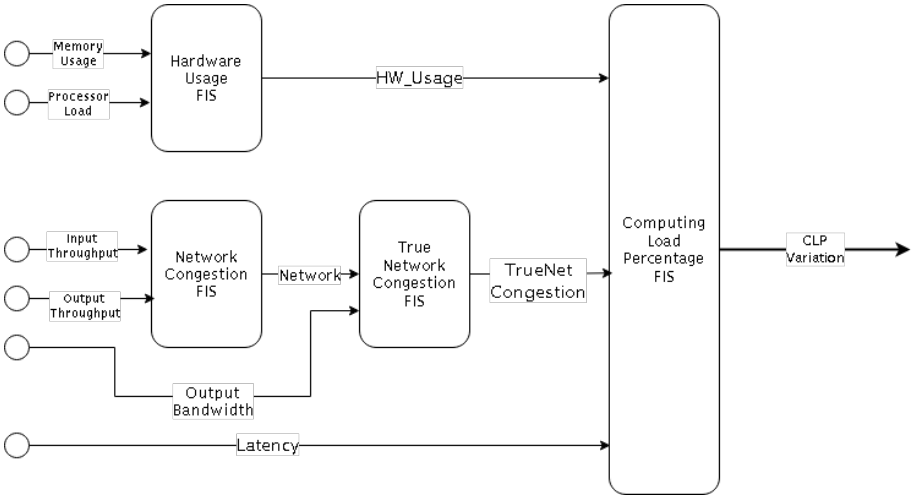

Figure 1: Fuzzy System Architecture

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.

Similiar 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:

Network Congestion FIS

True Network Congestion FIS

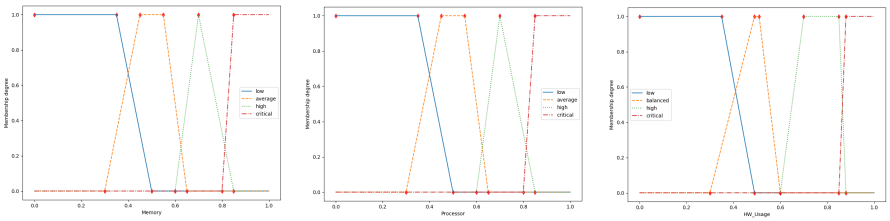

Figure 2: Membership Functions for Hardware Input and Output variables

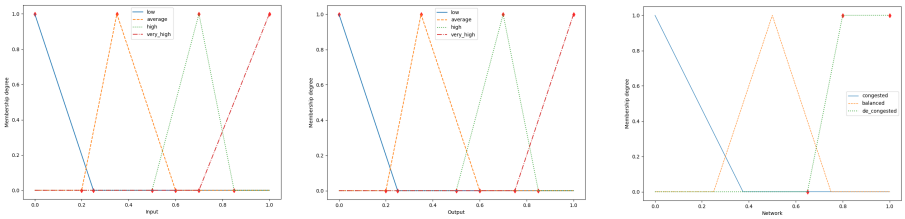

Figure 3: Membership Function for Net. Congestion Input and Output variables

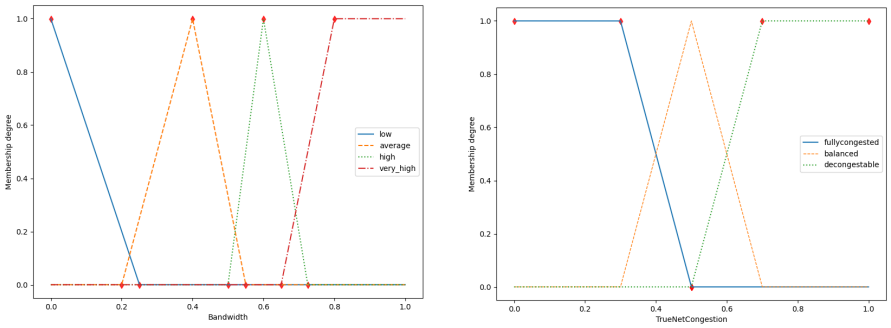

Figure 4: Membership Function for True Net. Congestion Input and Output variables

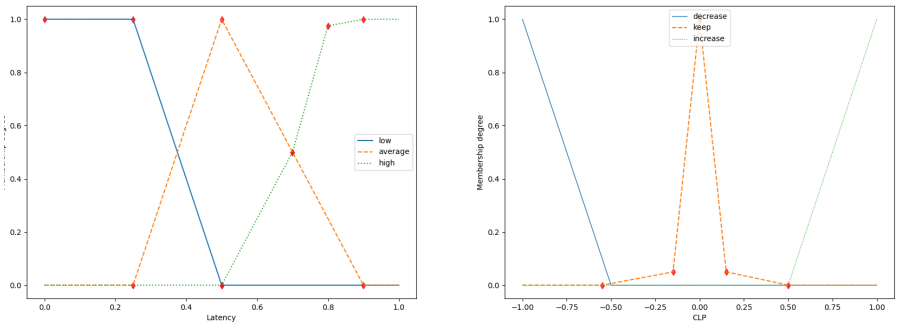

Figure 5: Membership Function for CLP-exclusive Input and Output variables

Final CLP FIS

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.

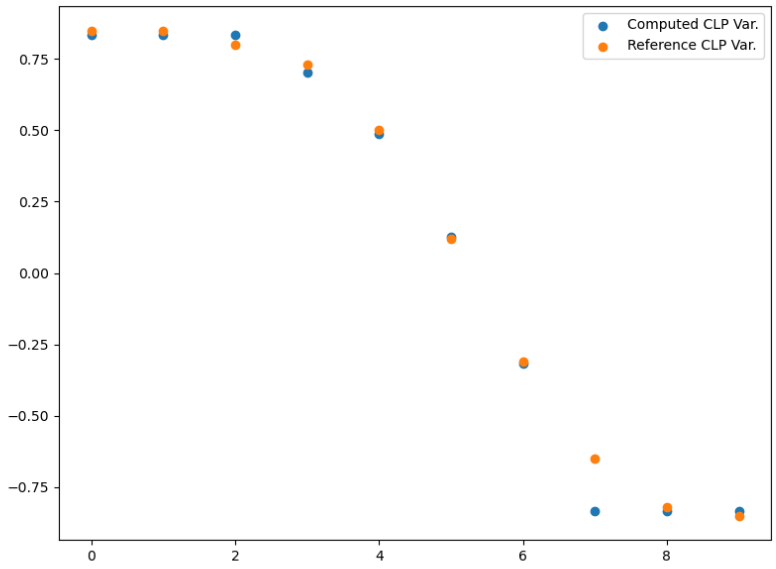

Figure 6: Computed CLPv compared to reference CLPv using initial dataset

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.

Metrics

Conclusions