Latent Space Encoding for Interpretable Fuzzy Logic Rules in Continuous and Noisy High-Dimensional Spaces

John Wesley Hostetter and Min Chi
College of Engineering Department of Computer Science
North Carolina State University, Raleigh, NC, USA

***Abstract*—This study introduces a general approach for generating fuzzy logic rules in regression tasks with complex, high-dimensional input spaces. The method leverages the power of encoding data into a *latent* space, where its uniqueness is analyzed to determine whether it merits the distinction of becoming a noteworthy exemplar. The efficacy of the proposed method is showcased through its application in predicting the acceleration of one of the links for the Unimation Puma 560 robot arm, effectively overcoming the challenges posed by non-linearity and noise in the dataset.**

I. Introduction

Finding accurate and easily interpretable fuzzy logic rules is a common challenge in fuzzy modeling [1]. Linguistic fuzzy modeling prioritizes interpretability, while precise fuzzy modeling prioritizes accuracy [2]. Despite their different approaches, both strive to minimize the size of their knowledge base (i.e., collection of fuzzy logic rules) to achieve their goals [3]. Various techniques have been proposed in the literature to construct this knowledge base. In techniques such as the Wang-Mendel (WM) Method [4] or members of the pseudo-outer-product fuzzy neural network (POPFNN) family [5]–[12], the construction of the knowledge base is related to the concept of identifying *exemplars* to generate fuzzy logic rules. In other words, if a data point is exceptional, it should be linked to a rule so we can learn how to manage such exceptional cases. These methodologies have widespread adoption in fuzzy modeling and have shown great promise. However, such methods do not scale well as the problem's dimensionality increases, often referred to as "the curse of dimensionality". Specifically, the data distribution becomes more sparse as the number of dimensions grows, and nearly every data point appears to be an exemplar in high dimensions. This is problematic since approaches such as the WM Method or RSPOP (a member of POPFNN) have a theoretical worst-case scenario of generating fuzzy logic rules that grow linearly with respect to the training data size. Such performance guarantees are unacceptable in situations with extensive data observations and many attributes.

This brings us to the primary motivation for our paper. Suppose we could convert or encode our high-dimensional data into a lower-dimensional format that remained representative of the original data. In that case, we might be able to leverage the "tried and true" techniques such as WM Method, even for high-dimensional problems.

We present a general algorithm designed to tackle the challenge of fuzzy modeling in high-dimensional data. First, we train an artificial neural network called an auto-encoder to learn an effective lower-dimensional encoding of the high-dimensional data. Then, we pair the original high-dimensional data with its respective lower-dimensional (latent) representation so that the pair remains and moves together through the subsequent process. Finally, we examine the latent encoding instead rather than consult the higher-dimensional (original) representation to identify whether this data observation should generate a fuzzy logic rule. Our intuition is that if a data point stands out as exceptional in the lower-dimensional (latent) representation, it is equally exceptional in the higher-dimensional (original) data space. Since the latent and original representations have been paired together, the two, in essence, move together *in lockstep* through the algorithm. For this reason, we call it "*The Latent Lockstep Method*".

In this paper, we illustrate the potential of the Latent Lockstep Method by applying it to a dataset [13] generated by a realistic simulation for the Puma 560 Arm [14]. We chose this benchmark for a few reasons. First, there are two versions with a different number of input features (8 attributes or 32 attributes), so we can see how the methods' performance changes as dimensionality increases. Second, the attributes (input and output) are continuous values. Third, with more than 8,000 data observations, this data is quite sizeable, so we can see if the number of fuzzy logic rules identified grows linearly concerning the training data size. Lastly, the dataset offers two versions of noise that have tainted the input values (moderate or high), effectively making the problem even more challenging. We show existing methods do not scale well as the problem difficulty increases, but our Latent Lockstep Method remains robust and effective in our experiments.

II. Methodology

A. Fuzzy set identification

We use a single-pass efficient algorithm, CLIP [15], to create Gaussian membership functions. It is inspired by the process of human category learning and is more flexible than other methods, such as Fuzzy C-Means [16], as it does not require the number of fuzzy sets to be predefined and may

be used online or offline. At a high level, CLIP creates a fuzzy set for the first data observation seen in the training data that covers the entire domain. The fuzzy set is defined by Gaussian membership functions, with parameters for the center and width. CLIP calculates the similarity match between the input value and existing fuzzy sets if a new data observation is encountered. The best-matched fuzzy set is used if the similarity exceeds a contrasting threshold. If not, a new fuzzy set accommodates the input value. CLIP continues to refine fuzzy sets as it processes training data.

Upon seeing the first data observation, $\mathbf{x}$, in the training data, X, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and n is the maximum dimensionality of the input space, CLIP will create a fuzzy set —that represents a *concept* [15] —which covers the entire domain for some input dimension i where $1 \leq i \leq n$ (i will always refer to the i^{th} input dimension). Since CLIP produces fuzzy sets that are defined by Gaussian membership functions [15], then this newly created fuzzy set has parameters $c_i^1 = x_i$ and $\sigma_i^1 = \Phi\left(\sqrt{-\frac{(\min_i - x_i)^2}{\log \epsilon}}, \sqrt{-\frac{(\max_i - x_i)^2}{\log \epsilon}}\right)$, where c_i^1 and σ_i^1 are the center and width of the Gaussian membership function that describes A_i^1, respectively (the superscript 1 is to emphasize that it is the first fuzzy set created in i). A newly created membership function is centered upon the presented value, while $\Phi(\sigma_i^j, \sigma_i^l) := \frac{1}{2}[\sigma_i^j + \sigma_i^l]$ defines a *regulator function* and $j \neq l$.

The regulator function ensures that each fuzzy set has a suitable buffer space around its center and retains its unique meaning. The minimum membership threshold ϵ is established to regulate the membership value within the domain $[\min_i, \max_i]$. The regulator function prevents malformed membership functions, especially when the center is close to the boundary's edge. It helps maintain a desirable Gaussian shape with an equal spread on both sides of the center.

If a fuzzy set already exists in i, but a new data observation $\breve{\mathbf{x}}$ has been encountered (i.e., $\breve{\mathbf{x}} \neq \mathbf{x}$), then CLIP will calculate a similarity match between the input value $\breve{x}_i$ and all existing fuzzy sets in i; this "similarity match" is $\mu_{A_i^j}(c_i^j, \sigma_i^j; \breve{x}_i)$ where A_i^j is the j^{th} fuzzy set to be created in i and c_i^j, σ_i^j are its Gaussian membership function's properties, respectively.

The best matched fuzzy set is $\star = \text{argmax}_j \mu_{A_i^j}(c_i^j, \sigma_i^j; \breve{x}_i)$ s.t. "$\star$" references the best matched existing fuzzy set in i. If the similarity between $\breve{x}_i$ and $A_i^\star$ exceeds a *contrasting threshold* κ, then this fuzzy set, $A_i^\star$, is able to represent $\breve{x}_i$ satisfactorily. Otherwise, a new fuzzy set will be created to accommodate for $\breve{x}_i$ while adjusting and refining fuzzy sets in i. Formally, this new fuzzy set in i is created by

$$c^{J_i(t)+1} = \breve{x}_i$$
$$\sigma^{J_i(t)+1} = \begin{cases} \sigma^L & \text{if } j_i^R = \text{NULL} \\ \sigma^R & \text{if } j_i^L = \text{NULL} \\ \Phi(\sigma^L, \sigma^R) & \text{otherwise} \end{cases} \quad (1)$$

where $J_i(t)$ is the number of fuzzy sets that have been created for i *thus far* at time-step t, and

$$\sigma^L = \Phi\left(\sqrt{-\frac{(c^{j_i^L} - \breve{x}_i)^2}{\log \epsilon}}, \sigma^{j_i^L}(t)\right) \quad (2)$$

$$\sigma^R = \Phi\left(\sqrt{-\frac{(c^{j_i^R} - \breve{x}_i)^2}{\log \epsilon}}, \sigma^{j_i^R}(t)\right). \quad (3)$$

After creating the new fuzzy set, the existing fuzzy sets in i accommodate this new addition; this ensures that the fuzzy sets within i remain distinct. For computational simplicity, only the newly created fuzzy set's left and right neighbors (if they exist) are adjusted/refined. This leads to three possibilities; the new fuzzy set has:

1) no left neighbor (i.e., $j_i^L = \text{NULL}$ by (4)), then the right neighbor is fixed: $\sigma^{j_i^R}(t+1) = \sigma^{J_i(t)+1}$ via (3)
2) no right neighbor (i.e., $j_i^R = \text{NULL}$ by (5)), then the left neighbor is fixed: $\sigma^{j_i^L}(t+1) = \sigma^{J_i(t)+1}$ via (2)
3) left & right neighbors (by (4) and (5)), then both are: $\sigma^{j_i^L}(t+1) = \sigma^{j_i^R}(t+1) = \sigma^{J_i(t)+1}$ via (2) & (3)

The following formulas are used to determine neighboring fuzzy sets' eligibility for modification:

$$j_i^L = \begin{cases} \text{NULL} & \text{if } c^{j_i} \geq \breve{x}_i \text{ for } 1 \leq j_i \leq J_i(t) \\ \arg\min_{c^{j_i} < \breve{x}_i} |c^{j_i} - \breve{x}_i| & \text{otherwise} \end{cases} \quad (4)$$

$$j_i^R = \begin{cases} \text{NULL} & \text{if } c^{j_i} \leq \breve{x}_i \text{ for } 1 \leq j_i \leq J_i(t) \\ \arg\min_{c^{j_i} > \breve{x}_i} |c^{j_i} - \breve{x}_i| & \text{otherwise.} \end{cases} \quad (5)$$

Repeat for $1 \leq i \leq n$ using every $\mathbf{x} \in X$.

B. The Wang-Mendel Method

The Wang-Mendel (WM) Method is a well-established technique for generating fuzzy logic rules and is widely used in the field [4], [17]–[19]. It follows a five-step process designed for supervised learning. Our task primarily focuses on the second step, which involves converting candidate fuzzy logic rules into their fuzzy representation. In the following, we have adapted the WM Method to *only* map input data to fuzzy sets in the input space. This is done for two reasons: (1) we use a zero-order Takagi-Sugeno-Kang (TSK) FLC with product-inference engine [3] and (2) we want our proposed method to be independent of supervised learning; specifically, we want to propose a fuzzy logic rule generation method that is capable of other tasks such as fuzzy reinforcement learning [20].

Given a set of training data, X, where $\mathbf{x} \in X$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the method transforms each $\mathbf{x}$ into its fuzzy representation through a Cartesian product of fuzzy sets, where each fuzzy set corresponds to a particular input dimension. To determine which fuzzy sets are in this Cartesian product, for $1 \leq i \leq n$ of $\mathbf{x}$, we select the fuzzy set that x_i attains the highest degree of membership to:

$$\star = \arg\max A_i^j(x_i) \text{ for } 1 \leq j \leq J_i \quad (6)$$

where J_i represents the number of fuzzy sets within the ith dimension. Unlike before, J_i is now a constant value as CLIP has been completed and is no longer a function of time t.

A fuzzy logic rule links a combination of fuzzy sets to a specific decision. However, for the sake of versatility, we will use the scalar $\mathbf{0}$ as the decision, as this can be learned through back-propagation and gradient descent algorithms. Thus, given a candidate $\mathbf{x}$, we generate a fuzzy logic rule in the form:

$$Rule_k : (A_1^\star, A_2^\star, \ldots, A_n^\star) \mapsto \mathbf{0} \tag{7}$$

where $\star$ satisfies (6) for $1 \leq i \leq n$ and $Rule_k$ means the k^{th} fuzzy logic rule ($k \geq 1$); rules with identical antecedents are eliminated to prevent redundancy in the knowledge base.

C. Encoding data to a latent space

Our Latent Lockstep Method for fuzzy logic rule generation utilizes an artificial neural network architecture known as an auto-encoder [21]. Unlike traditional neural networks that map input data to a set of labels as their predicted values, an auto-encoder encodes the input data and then decodes it to retrieve the original values. This method comprises two distinct parts: the encoder and the decoder. The encoder aims to learn a compact and effective representation of the input data, often in a lower or higher dimensional space. The decoder "unpacks" the encoded data, reversing the encoding process. This approach is akin to encryption, where the encoded representation of the input data is deciphered to retrieve the original information.

The intuition behind our approach is driven by the belief that the latent space, or the compressed data representations, is vital to accurately identifying the true exemplars. If a data point shines in the lower dimensional representation, it is a genuine standout and should be turned into a fuzzy logic rule. As outlined in the pseudo-code, the method is versatile, straightforward, and a natural evolution of the WM Method (referred to as "WM" in Algorithm 1). This seemingly small change leads to a substantial decrease in fuzzy rules, making the results more interpretable and easily comprehended.

This paper implemented the Latent Lockstep Method's encoder and decoder as artificial feed-forward neural networks with hyperbolic activation functions. In theory, any universal function approximator could be applied here. The encoder has two sequential hidden layers: the first has a (default) size equal to half of the input space's dimensionality, and the second has a size equal to the specified latent space dimensionality (a hyper-parameter in our method). Between these layers, hyperbolic activation is applied to the linear mappings to introduce non-linearity. The decoder has a similar architecture but in reverse ordering. Again, any neural architecture may be used so long as adequate performance is achieved in encoding and decoding; this architecture is simply a suggestion.

Algorithm 1 The Latent Lockstep Method

Input: input training data, X
Output: fuzzy logic rules

```
rules ← ∅
latent rules ← ∅
encoder, decoder ← train(X)
X_encoded ← encoder(X)

A_encoded ← CLIP(X_encoded)
A_decoded ← CLIP(X)
for each x_encoded ∈ X_encoded do
 new latent rule ← WM (x_encoded, A_encoded)
 if new latent rule ∉ latent rules then
    add new latent rule to latent rules
    rule ← WM(x, A_decoded)
    add rule to rules
```

III. Experiments & Results

We demonstrate the Latent Lockstep Method's performance compared to the WM Method [17] and Wang-Mendel with Evolving Clusters (WM-EC) Method [22] across select high-dimensional datasets with continuous input and output space. We evaluate its effectiveness (MSE), interpretability (fuzzy logic rule count), and robustness (MSE and fuzzy logic rule count fluctuations).

A. The Unimation Puma 560 Robot Arm

The Puma 560 Arm is an 8-Link All-Revolute Robot Arm, and the datasets [13] were collected by a realistic simulation of the forward dynamics using the Matlab Robotics Toolbox (Release 3) [14]. The task is to predict the distance of the end-effector from a target. The input variables include joints' positions and angles, links' lengths, etc. There are 8 versions of the data available with varying difficulties, such as the number of inputs, the non-linearity of the data, and the amount of noise (sampled uniformly) that has corrupted the input values. For more details regarding the datasets, please refer to the paper published by its author, as it is well-described [13].

Within each experiment scenario, the dataset was split according to a $60/20/20$ ratio such that 60% of the data ($4,888$ data observations) was used for training, 20% ($1,638$ data observations) was for validation, and 20% ($1,639$ data observations) of the data was reserved for testing.

1) 8 continuous inputs & moderate noise: We begin with a version of this dataset available on OpenML called *kin8nm*; it has nine continuous features: 8 are the inputs (same types as mentioned previously), and 1 is the target. The dataset is described as highly non-linear and containing moderate noise. Although this data is not necessarily high-dimensional, this benchmark aims to show that the WM Method fails to overcome problems with only eight dimensions.

The experiment settings are as follows: the parameters for CLIP are $\epsilon = 0.2$ and $\kappa = 0.7$; the learning rate is $\eta = 3e-3$, the batch size is 128, and training occurs for 20 epochs. The WM Method has no parameters specific to it. The WM-EC Method has a distance threshold, and its search space was restricted to $[0.5, 1.0]$ for this experiment. The Latent Lockstep Method uses a hyper-parameter that controls the dimensionality of the latent space. Here, we allowed it to range from 1 to 4. We used a model-based approach called Tree Parzen Estimator [23] implemented by a Python library called Optuna. This Bayesian optimization technique uses sophisticated heuristics to continue the hyper-parameter search

TABLE I

FOR THE LATENT LOCKSTEP METHOD, FUZZY LOGIC RULES AND THEIR VALIDATION LOSS ARE SHOWN AS MEAN AND STANDARD DEVIATION (IN PARENTHESES) FOR EACH ATTEMPTED LATENT SPACE DIMENSIONALITY.

Dimensionality	Rules Count	Validation Loss
1 (14 trials)	3.5 (0.5)	0.071 (0.006)
2 (13 trials)	11.46 (2.499)	0.063 (0.011)
3 (10 trials)	39.2 (5.724)	0.059 (0.012)
4 (13 trials)	106.15 (14.733)	0.062 (0.010)

in areas that achieve the best results. We ran 50 trials for each method to see how the algorithms performed.

As anticipated, the WM Method was inadequate in generating a reasonable number of fuzzy logic rules for the dataset. It consistently identified an excessive 3,036 fuzzy logic rules, greatly exceeding our computational abilities to calculate the FLC outputs. As a result, we are unable to report the losses here. The fact that the WM Method identified so many fuzzy logic rules represents a significant failure in achieving our primary objective of generating a minimal number of rules.

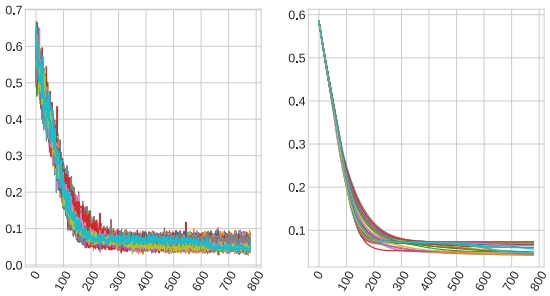

Fig. 1. The Latent Lockstep Method: (left) training loss; (right) validation loss. The x-axis is the batches.

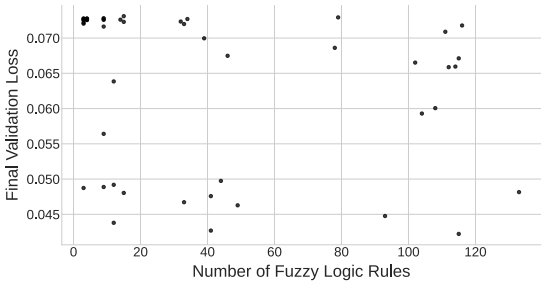

Fig. 2. Using the Latent Lockstep Method on 8 continuous input data, each trial is represented by a marker to display the relationship between the number of fuzzy logic rules and the FLC's final validation loss (lower is better).

Fig. 1 shows FLC's training and validation loss for each Latent Lockstep Method trial. Fig. 2 displays the possible fuzzy logic rules, ranging from 3 to 133, and validation loss, ranging from 0.042 to 0.073. Table I breaks down trials to observe how the latent space's dimensionality affects rule count and validation loss. The best model's test loss is 0.042.

Fig. 3 displays FLC's training and validation loss for each WM-EC Method trial. As shown in Fig. 4, possible fuzzy logic

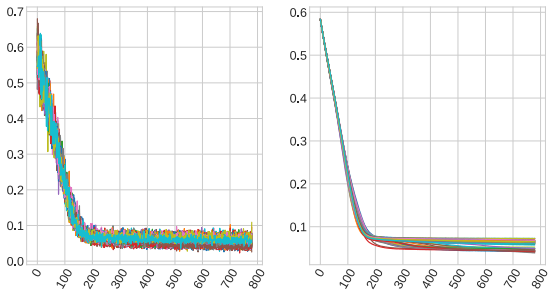

Fig. 3. The WM-EC Method: (left) training loss; (right) validation loss. The x-axis is the batches.

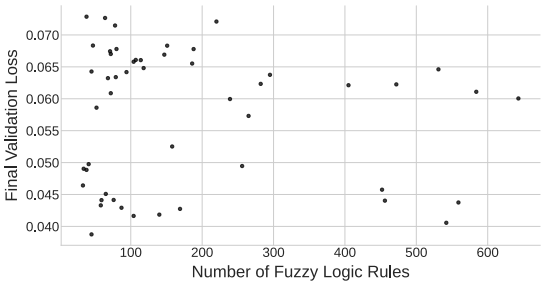

Fig. 4. Plotting each trial as a marker, the WM-EC Method was used on 8 continuous input data to show the relationship between the number of fuzzy logic rules and the FLC's final validation loss (lower is better).

rules range from 33 to 643, with validation loss ranging from 0.039 to 0.073. Table II breaks down the trial runs to examine how latent space dimensionality affects fuzzy logic rules and validation loss. The best model (concerning rule count and validation loss) has a test loss of 0.046.

The WM Method was ineffective in generating a reasonable number of fuzzy logic rules in the 8-dimensional example. The Latent Lockstep Method (across all 50 trials) generates an average of 39.4 (std. dev. $= 42.27$) rules, whereas the WM-EC Method (across all 50 trials) generates an average of 185.66 (std. dev. $= 171.2$) rules. According to an independent samples t-test, this difference is statistically significant with $t(49) = 5.806$ and a p-value ≤ 0.0005.

2) 32 continuous inputs & high noise: The WM Method is unsuitable for high-dimensional problems, as was quickly witnessed in the previous experiment. However, the WM-EC Method was at least competitive with the proposed Latent

TABLE II

FOR THE WM-EC METHOD, FUZZY LOGIC RULES AND THEIR VALIDATION LOSS ARE SHOWN AS MEAN AND STANDARD DEVIATION (IN PARENTHESES) FOR EACH VALUE BIN OF THE DISTANCE THRESHOLD.

Distance Threshold	Rules Count	Validation Loss
$[0.5, 0.6]$ (9 trials)	516 (71.068)	0.054 (0.009)
$[0.6, 0.7]$ (9 trials)	233.33 (42.599)	0.060 (0.009)
$[0.7, 0.8]$ (10 trials)	123.7 (21.895)	0.060 (0.010)
$[0.8, 0.9]$ (14 trials)	70.07 (9.384)	0.058 (0.011)
$[0.9, 1.0]$ (8 trials)	40.13 (4.910)	0.055 (0.011)

Lockstep Method. We will now demonstrate how this apparent equivalence may be misleading by increasing the input dimensionality to 32 and increasing the amount of noise within those inputs to demonstrate that not even the WM-EC Method is robust to scaling the input dimensionality.

The experiment settings are the same as before, where the parameters for CLIP are $\epsilon = 0.2$ and $\kappa = 0.7$, the learning rate is $\eta = 3e - 3$, the batch size is 128, and training occurs for 20 epochs. Also, instead of allowing the dimensionality of the latent space to fluctuate (in the case of the Latent Lockstep Method) or the distance threshold to change (in the case of the WM-EC Method), we selected the best hyperparameters for those based upon the eight continuous inputs and moderate noise in an attempt to see *how robust* or *how sensitive* the methods are to a change in input dimensionality or noise frequency. So, the dimensionality of the latent space was determined to be 2, and the distance threshold was 1.0.

Despite the WM-EC Method's strong performance in the previous scenario, it failed to generalize to the increase in dimensionality and noise. It consistently identified 4,888 fuzzy logic rules, which exceeded our computational abilities to calculate the FLC outputs. Still, during a single run that was able to finish, the measured loss was acceptable at 0.0008, but this is effectively a k-nearest neighbors model as it has "lazily memorized" the training data (there were 4,915 data observations) and their associated outputs.

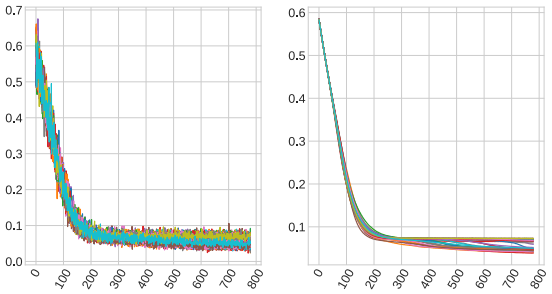

Fig. 5. The Latent Lockstep Method: (left) training loss; (right) validation loss. The x-axis is the batches.

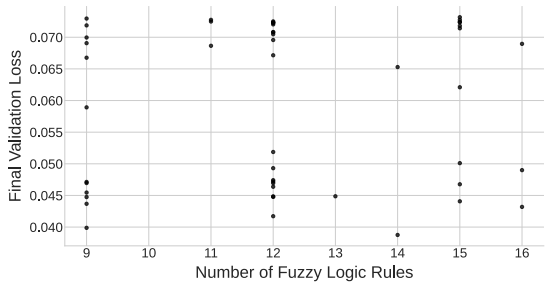

Fig. 6. Plotting each trial as a marker, the Latent Lockstep Method was used on 32 continuous input data to show the relationship between the number of fuzzy logic rules and the FLC's final validation loss (lower is better).

Fig. 5 displays FLC's training and validation loss for each Latent Lockstep Method trial. Fig. 6 shows that the method can use 9 to 16 fuzzy logic rules, achieving a validation loss of approximately 0.039 to 0.073, with an average of 12 rules (std. dev. = 2.27). The best trial used only nine rules, with a validation loss of 0.039 and a test loss of 0.044.

IV. Related Work

In the early 1990s, research on self-organizing FLCs and fuzzy modeling led to several breakthroughs. One notable advancement was Lin and Lee's connectionist representation of an FLC, which allowed for automatic design through a combination of unsupervised and supervised learning, similar to an artificial neural network [24]–[26]. Kohenen's feature-maps algorithm identified fuzzy sets and considered all possible fuzzy logic rules. Although, this approach does not scale well with increasing input dimensionality. These difficulties led to the study of FLC's equivalence with artificial neural networks [27]; such methods were successfully used in Iris classification [27], LED display recognition [28], and intelligent tutoring systems [29]. Still, their rules are conditioned upon neuron activations. A sub-category of neuro-fuzzy networks, the POPFNN family [5]–[12] improved upon previous methods. POPFNN-TVR [5] used a single pass to identify fuzzy logic rules with POP learning. However, its knowledge base could still snowball. To address this issue, LazyPOP [6] debuted to reduce knowledge base growth but required user-defined thresholds. Subsequently, POPFNN-CRI [8] was introduced, eliminating the need for supervised learning but is frequently employed for further refinement.

To provide a rough comparison, the performance of POPFNN-CRI was evaluated on the Iris dataset. It required 42 fuzzy logic rules to solve the 4-input dimension problem with 150 data observations [8]. Later, SANFIS was developed and solved the Iris dataset with just 3 fuzzy logic rules [30]. The algorithm was also applied to the Nakanishi datasets [31], which consist of data describing a non-linear system (4 attributes; 50 data observations), the human operation of a chemical plant (5 features; 70 data observations), and the daily price of a stock in the stock market (10 attributes, 100 data observations). The POPFNN-CRI algorithm identified 192, 1,920, and 3,000,000 fuzzy logic rules for the above datasets [9]. To improve this, RSPOP-CRI was proposed [9], identifying only 22, 24, and 50 fuzzy logic rules, respectively. After attribute and rule reduction, RSPOP-CRI settled upon 17, 14, and 29 rules [9]. The rule identification of RSPOP-CRI is superior to LazyPOP since it has linear fuzzy logic rule growth concerning the training data. Some of the latest and most cutting-edge research still relies on the RSPOP approach, such as ieRSPOP [11], ARPOP [10], and PIE-RSPOP [12].

Recent literature has explored the use of auto-encoders in fuzzy rule reduction, but the resulting fuzzy logic rules' premises are no longer conditioned upon the original features; instead, they are built directly from the latent representations, which hampers interpretability concerning semantic meaning [32]. In our Latent Lockstep Method, we propose using the latent representation *to identify* exemplary data points rather than convert the data to a lower-dimensional representation

to address the "rule explosion" problem. For example, a proposed Deep Learning Based Fuzzy Classifier leveraged a β-Variational Autoencoder for high-dimensional data [33]; still, the disentangled latent space representation learned is then used as the semantics for the fuzzy logic rules, and the interpretability of the rule base was constrained to analyzing latent traversals and latent dimensions' heat maps. Similarly, fuzzy auto-encoders have constructed hierarchical FLCs layer-by-layer [34], but we are interested in "flat" FLCs in this work.

Genetic fuzzy systems for fuzzy linguistic modeling have also explored high-dimensional regression problems. For example, embedded genetic database learning (involved variables, linguistic terms, etc.) using multi-objective evolutionary algorithms may produce a set of fuzzy logic rules with reduced model complexity. Such algorithms include, for example, METSK-HD^e [35], FSMOGFS^e + TUN^e [36], MOFFS [37], FMIFS [38], [39], MOFS^{CE} [40], and MOKBL+MOMs [37], which have been evaluated on the Unimation Puma 560 robot arm task as well, resulting in an average number of fuzzy rules ranging from 13.8 [37] to 87.5 [35]. Rule identification with the Latent Lockstep Method may appear approximately comparable to these existing works, but the advantage is its implementation simplicity and computational time. Algorithms like MOKBL+MOMs require complex, multi-step procedures to identify and prune fuzzy logic rules with appropriate fitness function definitions. These techniques generate and evaluate several candidate FLCs, whereas the Latent Lockstep Method generates only a single FLC. As a result, executing MOKBL+MOMs may take upwards of two hours on average for a single run [37], compared to the Latent Lockstep Method, which only takes a few minutes to train the auto-encoder. Lastly, the Latent Lockstep Method's minimal assumptions regarding fuzzy set definition or objective (e.g., regression, classification) permit it additional flexibility and adaptability to various settings, such as fuzzy reinforcement learning [20], where actions' Q-values may not be known in advance. Future work could explore incorporating auto-encoders into the rule identification process of genetic fuzzy systems to improve performance, and computational time, or leverage the mechanisms introduced in these works to eliminate unnecessary antecedents in the fuzzy logic rules' premises.

V. Limitations

Validation and comparison to existing methods are constrained by two limitations: (1) the design, as well as analysis of this algorithm, is late-breaking, and (2) to the best of our knowledge, most existing works in fuzzy logic rule generation often claim to work in high dimensions but are only benchmarked on space with a dimensionality of 10 or less [9], [41]–[43]. We go beyond these dimensions here, so most existing methods cannot serve as a baseline as their use is computationally intractable or prohibitive concerning computational time. In future research, comparing the Latent Lockstep Method to evolutionary or genetic algorithm approaches would be valuable. Although the latter often requires a predefined number of rules in the generation population, the Latent Lockstep Method automatically determines the necessary number for fuzzy logic control.

Despite using a PyTorch-based neuro-fuzzy network, loss plots could not be generated for FLCs with more than 3k rules. Training such models wasn't computationally feasible. However, the algorithms' main goal is to produce minimal fuzzy rules and then evaluate their performance. FLCs with large knowledge bases fail this first and essential criterion.

VI. Discussion and Broader Impact

This paper introduces the Latent Lockstep Method, a stable algorithm with tremendous potential for generating fuzzy logic rules in high-dimensional spaces as it aims to reduce the required number of rules. It is task-indifferent and could be used for classification or fuzzy reinforcement learning [20] since it only relies upon the input data for rule generation.

The Latent Lockstep Method offers an additional advantage over ad hoc design methods like the WM-EC Method. It enables the evaluation of the effectiveness of identified fuzzy logic rules in advance. In contrast, the WM-EC Method requires a distance threshold to be specified, leading to uncertainty about the aggregation level's aggressiveness in the initial FLC design. The Latent Lockstep Method offers a degree of certainty by demonstrating its ability to decode the input data from its latent representation.

Lastly, the Latent Lockstep Method offers a new perspective on fuzzy logic rule generation by shifting the focus onto a lower dimensional encoding that may have theoretical implications regarding the growth of the FLC knowledge bases. For example, FLCs constructed by WM Method have a worst-case scenario of generating a knowledge base that grows linearly concerning the unique training data [4]. In the worst case, the Latent Lockstep Method grows linearly with unique *encoded* training data. Suppose the unique encoded training data is smaller than the unique training data. In that case, our method will produce FLCs with smaller knowledge bases than most existing works [4]–[12]. This offers new opportunities to investigate the connection between fuzzy logic rule generation and auto-encoders in fuzzy modeling research.

While there may still be room for improvement, particularly in reducing the generated rules' width (i.e., the number of antecedents/premises) and making them more human-readable, the Latent Lockstep Method represents a significant step forward in fuzzy linguistic modeling. Future work may explore its integration with Rough Set Theory, which has been shown to reduce the size of FLC knowledge bases [9]–[12].

VII. Acknowledgment

This research was supported by the NSF Grants: Integrated Data-driven Technologies for Individualized Instruction in STEM Learning Environments (1726550), CAREER: Improving Adaptive Decision Making in Interactive Learning Environments (1651909), and Generalizing Data-Driven Technologies to Improve Individualized STEM Instruction by Intelligent Tutors (2013502).

REFERENCES

[1] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 748–768, May 2000.

[2] R. Babuska, "Data-Driven Fuzzy Modeling: Transparency and Complexity Issues," Feb. 2001.

[3] J. Casillas, O. Cordón, F. H. Triguero, and L. Magdalena, *Interpretability Issues in Fuzzy Modeling*. Springer, Jun. 2013.

[4] L.-X. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, Nov. 1992.

[5] R. W. Zhou and C. Quek, "POPFNN: A Pseudo Outer-product Based Fuzzy Neural Network," *Neural Networks*, vol. 9, no. 9, pp. 1569–1581, Dec. 1996.

[6] R. Zhou and C. Quek, "A pseudo outer-product based fuzzy neural network and its rule-identification algorithm," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 2, Jun. 1996, pp. 1156–1161 vol.2.

[7] C. Quek and R. Zhou, "POPFNN-AAR(S): a pseudo outer-product based fuzzy neural network," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 29, no. 6, pp. 859–870, Dec. 1999.

[8] K. Ang, C. Quek, and M. Pasquier, "POPFNN-CRI(s): Pseudo Outer Product Based Fuzzy Neural Network Using the Compositional Rule of Inference and Singleton Fuzzifier," *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 33, pp. 838–49, Feb. 2003.

[9] K. Ang and C. Quek, "RSPOP: Rough Set-Based Pseudo Outer-Product Fuzzy Rule Identification Algorithm," *Neural computation*, vol. 17, pp. 205–43, Feb. 2005.

[10] E. Y. Cheu, C. Quek, and S. K. Ng, "ARPOP: An Appetitive Reward-Based Pseudo-Outer-Product Neural Fuzzy Inference System Inspired From the Operant Conditioning of Feeding Behavior in Aplysia," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 317–329, Feb. 2012.

[11] R. T. Das, K. K. Ang, and C. Quek, "ieRSPOP: A novel incremental rough set-based pseudo outer-product with ensemble learning," *Applied Soft Computing*, vol. 46, pp. 170–186, Sep. 2016.

[12] A. R. Iyer, D. K. Prasad, and C. H. Quek, "PIE-RSPOP: A brain-inspired pseudo-incremental ensemble rough set pseudo-outer product fuzzy neural network," *Expert Systems with Applications*, vol. 95, pp. 172–189, Apr. 2018.

[13] C. Rasmussen, R. Neal, G. Hinton, D. van Camp, M.Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani, "The DELVE manual," *The University of Toronto*, 1996.

[14] P. Corke, "A robotics toolbox for matlab," *IEEE Robotics Automation Magazine*, vol. 3, no. 1, pp. 24–32, 1996.

[15] S. W. Tung, C. Quek, and C. Guan, "SaFIN: A Self-Adaptive Fuzzy Inference Network," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1928–1940, Dec. 2011.

[16] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers Geosciences*, vol. 10, no. 2, pp. 191–203, 1984.

[17] L.-X. Wang, "The WM method completed: a flexible fuzzy system approach to data mining," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 6, pp. 768–782, Dec. 2003.

[18] T. Rutkowski, K. Łapa, and R. Nielek, "On Explainable Fuzzy Recommenders and their Performance Evaluation," *International Journal of Applied Mathematics and Computer Science*, vol. 29, no. 3, pp. 595–610, Aug. 2019.

[19] J. Kim and N. Kasabov, "HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Networks*, vol. 12, no. 9, pp. 1301–1319, Nov. 1999.

[20] H. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 724–740, Sep. 1992.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," 1986.

[22] J. W. Hostetter, M. Abdelshiheed, T. Barnes, and M. Chi, "A self-organizing neuro-fuzzy q-network: Systematic design with offline hybrid learning," in *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

[23] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011.

[24] C. Lin and C. Lee, "Real-time supervised structure/parameter learning for fuzzy neural network," in *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*, Mar. 1992, pp. 1283–1291.

[25] C.-T. Lin and C. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.

[26] ——, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 1, pp. 46–63, Feb. 1994.

[27] E. Kolman and M. Margaliot, "Are artificial neural networks white boxes?" *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 844–852, 2005.

[28] ——, "Knowledge Extraction From Neural Networks Using the All-Permutations Fuzzy Rule Base: The LED Display Recognition Problem," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 925–931, 2007.

[29] J. W. Hostetter, M. Abdelshiheed, T. Barnes, and M. Chi, "Leveraging fuzzy logic towards more explainable reinforcement learning-induced pedagogical policies on intelligent tutoring systems," in *2023 IEEE International Conference on Fuzzy Systems*. IEEE, 2023.

[30] J.-S. Wang and C. Lee, "Self-adaptive neuro-fuzzy inference systems for classification applications," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 6, pp. 790–802, Dec. 2002.

[31] H. Nakanishi, I. Turksen, and M. Sugeno, "A review and comparison of six reasoning methods," *Fuzzy Sets and Systems*, vol. 57, no. 3, pp. 257–294, 1993.

[32] R. K. Sevakula and N. K. Verma, "Fuzzy Rule Reduction using Sparse Auto-Encoders," in *2015 FUZZ-IEEE*, Aug. 2015, pp. 1–7.

[33] K. Bölat and T. Kumbasar, "Interpreting Variational Autoencoders with Fuzzy Logic: A step towards interpretable deep learning based fuzzy classifiers," in *2020 FUZZ-IEEE*, Jul. 2020, pp. 1–7, iSSN: 1558-4739.

[34] T. Zhao, H. Cao, and S. Dian, "A Self-Organized Method for a Hierarchical Fuzzy Logic System Based on a Fuzzy Autoencoder," *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 12, pp. 5104–5115, Dec. 2022.

[35] M. J. Gacto, M. Galende, R. Alcalá, and F. Herrera, "METSK-HDe: A multiobjective evolutionary algorithm to learn accurate TSK-fuzzy systems in high-dimensional and large-scale regression problems," *Information Sciences*, vol. 276, pp. 63–79, Aug. 2014.

[36] R. Alcala, M. J. Gacto, and F. Herrera, "A Fast and Scalable Multiobjective Genetic Fuzzy System for Linguistic Fuzzy Modeling in High-Dimensional Regression Problems," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 4, pp. 666–681, Aug. 2011.

[37] F. Aghaeipoor and M. M. Javidi, "MOKBL+MOMs: An interpretable multi-objective evolutionary fuzzy system for learning high-dimensional regression data," *Information Sciences*, vol. 496, pp. 1–24, Sep. 2019.

[38] M. Antonelli, P. Ducange, and F. Marcelloni, "Feature Selection Based on Fuzzy Mutual Information," in *Fuzzy Logic and Applications*, ser. Lecture Notes in Computer Science, F. Masulli, G. Pasi, and R. Yager, Eds. Cham: Springer International Publishing, 2013, pp. 36–43.

[39] M. Antonelli, P. Ducange, F. Marcelloni, and A. Segatori, "On the influence of feature selection in fuzzy rule-based regression model generation," *Information Sciences*, vol. 329, pp. 649–669, Feb. 2016.

[40] F. Jiménez, G. Sánchez, J. M. García, G. Sciavicco, and L. Miralles, "Multi-objective evolutionary feature selection for online sales forecasting," *Neurocomputing*, vol. 234, pp. 75–92, Apr. 2017.

[41] M. J. Gacto, R. Alcalá, and F. Herrera, "Handling High-Dimensional Regression Problems by Means of an Efficient Multi-Objective Evolutionary Algorithm," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*, Nov. 2009, pp. 109–114, iSSN: 2164-7151.

[42] A. A. Márquez, F. A. Márquez, and A. Peregrín, "An efficient multi-objective evolutionary adaptive conjunction for high dimensional problems in linguistic fuzzy modelling," in *2012 IEEE International Conference on Fuzzy Systems*, Jun. 2012, pp. 1–8, iSSN: 1098-7584.

[43] S. Ashraf and P. C. Shill, "Linguistic Fuzzy Modeling for High Dimensional Regression Problem Using Multi-Objective Genetic Algorithm," in *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*, Feb. 2018, pp. 1–4.