Evolver: An Evolving Neuro-Fuzzy System for Interpretable Long-Term Stock Market Forecasting

Miha Ožbot
Faculty of Electrical Engineering
University of Ljubljana
Ljubljana, Slovenia
miha.ozbot@fe.uni-lj.si

Igor Škrjanc
Faculty of Electrical Engineering
University of Ljubljana
Ljubljana, Slovenia
igor.skrjanc@fe.uni-lj.si

Vitomir Štruc
Faculty of Electrical Engineering
University of Ljubljana
Ljubljana, Slovenia
vitomir.struc@fe.uni-lj.si

***Abstract*—In the complex landscape of multivariate time series forecasting, achieving both accuracy and interpretability remains a significant challenge. This paper introduces the Evolving Transformer (Evolver), a novel recurrent neural network architecture combined with multi-head self-attention and fuzzy inference systems to analyze multivariate stock market data and conduct long-term time series forecasting. The method leverages LSTM networks and temporal attention to condense multivariate data into interpretable features suitable for fuzzy inference systems. The resulting architecture offers comparable forecasting performance to conventional models such as ARIMA and LSTM while providing valuable insights into information flow within the network. The method was examined on the real world stock market index S&P500. Initial results highlight both the promise and challenges of this approach, suggesting potential for further research and practical application in understanding and forecasting stock market behavior.**

***Index Terms*—Stock Market Prediction, LSTM, Multi-Head Attention, Fuzzy Systems, Interpretability, Deep Clustering**

I. Introduction

In stock price forecasts, the interpretability of the model is desired to explain the predictions and infer semantic meaning that can be understood by investors. While black-box models achieve greater accuracy, their predictions are hard to explain, which is very important in economics. One of the most influential advancements in the field of deep neural networks is attention-based architectures; they can offer some insight into the meaningful connections of the input signals [1] for technical analysis of the patterns from historical data. Conversely, the interpretability of the relation between inputs and outputs was one of the cornerstones of fuzzy systems [2]. We propose to combine recent advances in deep neural network time series forecasting [3], deep clustering [4], and evolving fuzzy systems [5] into a complex model, while maintaining enough transparency in the final layers to be interpreted by humans. The main contributions of this paper are as follows:

- Combining Long Short-Term Memory (LSTM) recurrent neural network architecture and the Multi-Head self-Attention (MHA) mechanism with Fuzzy Inference Systems (FIS) for multivariate multi-horizon time series forecasting (multi-step-ahead prediction).
- An unsupervised approach to deep unsupervised multivariate Gaussian clustering of the low dimensional latent space.

Evolving (Neuro-)Fuzzy Inference Systems (ENFIS) are models that can incrementally adapt both their parameters and structure online based on time-varying data. These models were initially proposed to address learning from data streams, where samples arrive online and are subject to slow conceptual drifts and abrupt shifts in data distribution, and large dimensionality of samples (curse of dimensionality [6]). The samples are generated online and usually cannot be stored but should be used to identify the evolving model and discarded immediately [7]. Evolving fuzzy and neuro-fuzzy models are commonly used for time series forecasting, specifically stock price prediction [8]. For example, a linguistic neuro-fuzzy model with a Hammerstein-Wiener type structure was used to predict stock price in [2]. An evolving Fuzzy Granular Predictor (eFGP) was used in [9] for cryptocurrency (Bitcoin) price prediction. An ensemble of evolving fuzzy models was used in [10] to predict the S&P500 time series with high accuracy. However, these methods only predict a few time steps in the future and use univariate data.

Interpretability is an important aspect of (evolving) fuzzy systems, resulting from complexity reduction and separation between the rule antecedent structure of the system and the associated linguistic and/or functional consequent part models [11]. In evolving systems, submodels are added, merged, split, and removed based on their similarity, usefulness, accuracy, etc., usually to maintain transparency as opposed to high accuracy. The main idea of the evolving fuzzy approach is to use the fuzzy model structure and online learning methodology to adapt to conceptual variations and gain insight into the rules that govern stock price movements. Some of the most popular evolving models are eTS+ [12], FLEXIFIX+ [13], eFuMo [14], eGauss+ [15], etc. A comprehensive survey of recent evolving fuzzy and neuro-fuzzy systems is conducted in [7].

Another crucial part of the fuzzy inference systems is the antecedent structures that define the activation or membership functions of the fuzzy rule. A common choice to describe antecedent data is multivariate Gaussian clusters, as they can

The authors acknowledge the financial support from the Slovenian Research and Innovation Agency – ARIS (program funding number P2-0219).

describe various data distributions or conditions under which the system operates. For this, an unsupervised or supervised clustering method is commonly employed to learn the means and covariance matrices used to describe the clusters. In this study, we explore the use of deep unsupervised clustering to train the proposed method end-to-end with backpropagation. One way to implement deep clustering is based on autoencoders to condense the data and the classic k-mean clustering method. A deep autoencoder with k-means clustering DEKM was proposed in [16]. The idea is that the input data is very high dimensional and clustering suffers from the curse of dimensionality. An encoder is therefore used to reduce the data dimensionality. Similarly, an autoencoder with k-means clustering of the latent space called fusion autoencoder FAE was proposed in [17] that uses dimensionality reduction and a compound loss to map inputs into a latent feature space that encourages feature separation for clustering. However, we would like to have soft cluster memberships for the fuzzy model antecedent. Another approach is to use variational autoencoder (VAE) deep clustering methods [4]. VAE are probabilistic graph based generative model that generate the latent space data based on the encoded information. However, in our case we want to be able to modify the clusters after training so a parametric representation is more adequate.

The most popular architecture for sequence-to-sequence modeling is the Long Short-Term Memory (LSTM), which is a type of recurrent neural network (RNN) that is very popular for any task involving long sequential data, e.g., text-to-speech, inference and regression, time series forecasting. A common approach in stock market prediction is to combine LSTM networks with attention-based layers for interpretability and to model long-term dependencies. The Temporal Fusion Transformer (TFT) [3] combines attention-based network structure, LSTMs, and gated residual networks (GRN) for multi-horizon forecasting of the S&P500 volatility. A study on several multi-horizon stock market forcasting with LSTM networks was conducted in [18]. However this was done for univariate data and for only 10 steps-ahead prediction. An LSTM network was used to predict the Indian stock market in [19]. While an interpretable attention-based variable selection for NASDAQ stock market prediction with an LSTM network was explored in [20] and a purely transformer based architecture for long-term time series forecasting was explored in [21]. Conversely, a method for day trading forecast of directional movement of S&P500 with LSTM (CuDNNLSTM) was presented in [22]. We propose a similar methodology by combining these advances with interpretable fuzzy systems.

The remainder of this paper is structure as follows. The methodology used in this study is presented in Section II. In Section III a multivariate multi-horizon case study and an ablation study are conducted. The results of the experiments are discussed in Section IV. Finally, the concluding remarks and future work are given in Section V.

II. Methods

Let $\underline{X}(k)=[\underline{x}(k-N),\ldots,\underline{x}(k)]\in\mathbb{R}^{N\times D_X}$ be the multivariate input data with D_X channels or input features, containing a main time series $\underline{Y}(k)=[y(k-N),\ldots,y(k)]\in\mathbb{R}^N$, and other multivariate input data up to a discrete time step k. Our objective is to predict the next H time steps of the main time series $\hat{\underline{Y}}(k)=[\hat{y}(k+1),\ldots,\hat{y}(k+H)]\in\mathbb{R}^H$. The proposed neuro-fuzzy system, termed the Evolver, is a sequence-to-sequence model comprised of the following main layers[1]:

1) An **LSTM** network [23] to encode long-term dependencies between time steps.
2) A **Multi-Head Self-Attention** network [1] to identify related time steps and enhance long-term information retention.
3) **Fully connected layers** to reduce the encoded data into two low-dimensional latent space representations.
4) A **fuzzy local model network** composed of multivariate Gaussian clusters [15] and ARIX local models [24] to generate the sequence forecast.

The Evolver architecture is presented in Figure 1.

A. Long-Short Term Memory Recurrent Neural Network

The multi-layer LSTM network was implemented as described in [25]. The hidden state (and cell state) of the LSTM network is initialized with zeros. This affects the first few time steps in the reconstructed output, but becomes irrelevant after 3–5 steps. The LSTM layer and the attention layer are used to reconstruct the input signal with a dense network at the output, functioning as a sequence-to-sequence encoder. This approach enables easier training of the attention layer that follows the LSTM network. Without this, the attention tends to focus on the entire LSTM output equally and at the beginning of the sequence. The LSTM can function as an input embedding for the Temporal multi-head attention that is used afterward [3]. Additionally, a cosine positional embedding [1] is utilized after the LSTM network. This is required because multi-head attention is computed in parallel, which removes the inherent order or position information.

B. Temporal Multi-Head Self-Attention

The Multi-Head Attention (MHA) mechanism, as proposed in [1], allows the model to focus on different parts of a sequence, by splitting the hidden features into several subspaces, computing an attention mechanism for each in parallel, and combining the outputs. Specifically, the input sequence $\underline{S}=[\underline{s}(1),...,\underline{s}(N)]\in\mathbb{R}^{N\times D_{in}}$ with length N and input dimension D_{in} is transformed with trainable parameter matrices $\underline{W}_K\in\mathbb{R}^{D_{in}\times D_h}$, $\underline{W}_Q\in\mathbb{R}^{D_{in}\times D_h}$, and $\underline{W}_V\in\mathbb{R}^{D_{in}\times d_{out}}$ into keys $\underline{K}=\underline{S}\,\underline{W}_K\in\mathbb{R}^{N\times D_h}$, queries $\underline{Q}=\underline{S}\,\underline{W}_Q\in\mathbb{R}^{N\times D_h}$, and values $\underline{V}=\underline{S}\,\underline{W}_V\in\mathbb{R}^{N\times d_{out}}$. A Scaled Dot-Product Attention is then computed as [1]:

$$\mathrm{Attention}(\underline{Q},\underline{K},\underline{V})=\mathrm{softmax}\Big(\frac{\underline{Q}\underline{K}^\top}{\sqrt{D_h}}\Big)\underline{V} \tag{1}$$

[1]The Evolver code is available at https://github.com/mihaozbot/Evolving-transformer

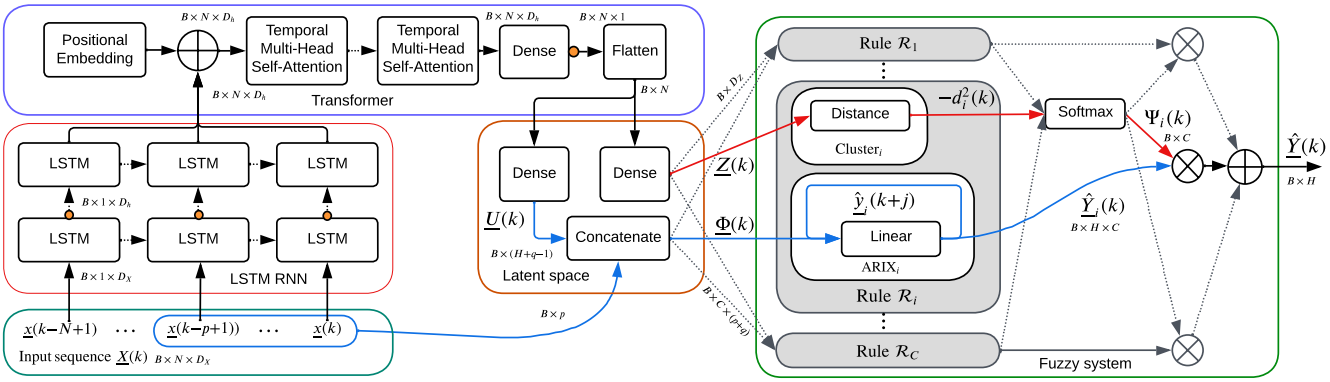

Figure 1. The Evolver architecture. The orange dots at the outputs of the LSTM and Dense layers represent the dropout mechanisms. The red and blue connections illustrate the data flow of the antecedent latent space clustering samples and the local model regression variables, respectively. The variable B is the batch size.

In multi-head attention, this is done multiple times in parallel, where each layer is called a head $h{=}1, ..., N_h$. The outputs of all heads are concatenated and reshaped with a linear transformation [26]:

$$\text{Multihead}(\underline{S}) = [\underline{H}^1, ..., \underline{H}^{N_h}]\underline{W}_O \tag{2}$$

$$\underline{H}^h = \text{Attention}(\underline{S}\,\underline{W}^h_{\text{q}}, \underline{S}\,\underline{W}^h_K, \underline{S}\,\underline{W}_V) \tag{3}$$

where $\underline{W}_O{\in}\mathbb{R}^{(N_h d_{out})\times D_{out}}$ is the output projection. We used the same hidden dimensions as proposed in [1], i.e, $D_h{=}d_{out}{=}D_{in}/N_h$. which results in the distribution of the D_{in} features of the input sequence into N_h sub-spaces.

We implemented a shared parameter matrix W_V for all heads based on the interpretable multi-head attention proposed in [3]. While this reduces the number of parameters, it allows for greater interpretability of the attention weights, i.e., the output is the same, just the attention of each head is different. Self-attention refers to the keys, queries, and values originating from the same input sequence $\underline{S}$. Additionally, the name temporal attention in this case refers to the attention being computed for the whole input sequence in order to find time-dependent connections between time steps as opposed to spatial attention, which learns relationships between features within a given time step.

C. Evolving Fuzzy System

The fuzzy Takagi-Sugeno model consists of multiple fuzzy rules with membership functions in the antecedent and an affine function in the consequent part of each rule. A fuzzy Takagi-Sugeno rule with one membership function in the antecedent and an affine linear function in the consequent can be described as follows:

$$R_i: \quad \text{IF} \quad \left(\underline{Z}(k) \sim \mathcal{Z}_i\right) \quad \text{THEN} \quad \underline{\hat{Y}}_i(k), \tag{4}$$

where $\sim$ denotes a soft membership to the fuzzy set $\mathcal{Z}_i$, $i{=}1, 2, ..., C$ is the index of the fuzzy rules, $j{=}1, ..., H$ is the future time step, $\underline{Z}(k){\in}\mathbb{R}^{D_Z}$ is the antecedent input vector, $\underline{\hat{Y}}_i(k){=}[\hat{y}_i(k{+}1), ..., \hat{y}_i(k{+}H)]$ is the output of the fuzzy rule.

The antecedent fuzzy sets $\mathcal{Z}_i$ can be represented by multivariate Gaussian clusters that can be rotated arbitrarily to define correlations between variables, since they can approximate a variety of data distributions [15]

$$d_i^2(k) = (\underline{Z}(k){-}\underline{\mu}_i)^\top \underline{\Sigma}_i^{-1}(\underline{Z}(k){-}\underline{\mu}_i), \tag{5}$$

where $d_i^2(k)$ is the Mahalanobis distance, $\underline{\mu}_i{\in}\mathbb{R}^{D_Z}$ and $\underline{\Sigma}_i{\in}\mathbb{R}^{D_Z\times D_Z}$ are the center and the covariance matrix of the cluster associated with the fuzzy rule $\mathcal{R}_i$. The membership functions are normalized with a softmax function $\Psi_i(k){=}\mathrm{e}^{-d_i^2(k)}/\sum_i^C \mathrm{e}^{-d_i^2(k)}{\in}[0,1]$, so that a unit partition is obtained $\sum_{i=1}^c \Psi_{ki}{=}1$. Finally, the output of the evolving neuro-fuzzy model is aggregated from all activated rules as $\underline{Y}(k){=}\sum_{i=1}^C \Psi_i(k)\underline{\hat{Y}}_i(k){\in}\mathbb{R}$.

In our case, we used the neural network encoder to generate the (non-)exogenous inputs $\underline{U}(k){=}[u_1, ..., u_p]{\in}\mathbb{R}^p$ for the ARIX model and the latent space vector $\underline{Z}(k)$ for the antecedent membership functions. The forecast of the ARIX local model of the rule $\mathcal{R}_i$ for the sequence $\underline{Y}(k)$ is then defined as:

$$\hat{y}_i(k+j) = (1 - A_i(\mathrm{q}^{-1}))\Delta y(k+j) + B_i(\mathrm{q}^{-1})u(k+j) \tag{6}$$

where $A_i\left(\mathrm{q}^{-1}\right){=}1{+}a_1\mathrm{q}^{-1}{+}...{+}a_p\mathrm{q}^{-p}$ is the auto-regressive (AR) polynomial, $B_i\left(\mathrm{q}^{-1}\right){=}b_1\mathrm{q}^{-1}{+}...{+}b_q\mathrm{q}^{-q}$ is the exogenous polynomial, q^{-1} is a discrete delay operator, i.e. $y(k{-}p){=}\mathrm{q}^{-p}y(k)$, p is the order of the auto-regressive model and q is the order of the exogenous input, and Δ is the differentiation operator with order d. The model starts by using the known input time series values $\underline{X}(k)$ and then adds the recursively computed predictions $y_i(k+j)$ in a sliding-window way for all j, resulting in a combined forecast $\underline{\hat{Y}}_i(k){=}[\hat{y}_i(k{+}1), ..., \hat{y}_i(k{+}H)]$.

D. Unsupervised deep clustering

Combining fuzzy models with deep neural networks presents some challenges. The main problem is that the Gaussian clusters are trained based on the latent space of

the network, which is also being trained and changing. The encoder neural network generates latent features that are then clustered in an unsupervised way with multivariate Gaussian clusters. It is possible to use a separate method for clustering the latent space features, such as Gustafson-Kessel, Fuzzy C-means, Gaussian Mixture Models, and additional clustering after training has also been shown to be beneficial [17]. However, this leads to behavior where the latent space samples reposition away from the clusters when backpropagation training is resumed, as it is usually much easier to train the latent space values than the Gaussian cluster centers and covariance matrices to improve the forecasting loss. Since in our case, the latent space data is also not labeled, we propose an unsupervised deep clustering approach. Several loss functions can be combined to enable good parameter convergence, while the space features should be easily separable for balanced cluster representation and interpretability of the structure. Using only a clustering loss does not ensure separation of the clusters, as all clusters may converge to similar mean values and start to overlap.

The multivariate Gaussian clusters were defined with the centers $\underline{\mu}_i$ and covariance matrices $\underline{\Sigma}_i$ as random parameter tensors. They were initialized with random values and then multiplied with scaling factors based on the initial spread of the latent space features. The latent space representation can be susceptible to overfitting due to a low number of rules in the fuzzy system, such that the validation and test latent space data differ greatly from the latent space during training. Dropout plays an important role in this regard. Notably, the latent space can look dramatically different with the training data and the test data if regularization measures, like dropout, are not used.

E. Training losses

The loss functions used in the proposed multi-task neural network can be separated into two groups: the forecast accuracy loss and the latent space clustering losses. One of the fuzzy layer rules (local model and clusters) may be more sensitive to changes in the shared bottom layers, and this imbalance can cause the overall model to perform poorly. A careful balance of the loss functions is required to maintain high accuracy and interpetability at the same time.

To train the multi-horizon time series forecast, we employ the common Mean Squared Error (MSE) loss function, defined as:

$$\mathcal{L}_{\text{MSE}} = \sum_{k} ||\underline{Y}(k) - \hat{\underline{Y}}(k)||^2, \tag{7}$$

We used a "winner-takes-all" local optimization for the consequent local models to improve the interpretability of each local model, as opposed to the standard global optimization. This was done by computing the forward pass only for the rule with the highest activation, which forces each rule to have a good output locally. Conversely, the intended fuzzy output was used to validate and test the model.

While the loss used in the Fuzzy C-means (FCM) clustering method [27] was used for deep unsupervised clustering, it is defined as:

$$\mathcal{L}_{\text{FCM}} = \sum_{i}^{C} \sum_{k} \text{softmax}(-d_i^2(k)) ||\underline{Z}(k) - \underline{\mu}_i||^2, \tag{8}$$

Two clustering regularization losses were used in the proposed approach. These losses improve the stability of the deep clustering method and also increase the interpretability of the model.

First, in order to keep the clusters distinguishable, a regularization loss is used to ensure cluster separation. Overlapping multivariate Gaussian clusters can be detected with the Bhattacharyya distance [11]:

$$\begin{aligned} d_B(m,n) = &\frac{1}{8}\big(\underline{\mu}_m - \underline{\mu}_n\big)^\top \left(\frac{\underline{\Sigma}_m + \underline{\Sigma}_n}{2}\right)^{-1} \big(\underline{\mu}_m - \underline{\mu}_n\big) + \\ &\frac{1}{2}\ln\left(\frac{\det\big(\frac{1}{2}(\underline{\Sigma}_m + \underline{\Sigma}_n)\big)}{\sqrt{\det\underline{\Sigma}_m \det\underline{\Sigma}_n}}\right), \end{aligned} \tag{9}$$

where $m{=}1, 2, ..., C$ and $n{=}1, 2, ..., C$ are the indexes of the two compared clusters for $m{\neq}n$, and $d_B(m,n){=}d_B(n,m)$. We can then formulate the overlapping regularization loss as:

$$\mathcal{L}_O = \sum_{m}^{C} \sum_{n \neq m}^{C} \frac{1}{d_B(m,n)} \tag{10}$$

Importantly, we detached the cluster variances Σ_i from the gradient computation, so that only the means μ_i are affected. This is because the Bhattacharyya distance returns a high value, i.e., the clusters are dissimilar or not overlapping, if the multivariate Gaussian distributions that represent the clusters have very dissimilar principal components in orientation and size. This can result in overlapping clusters with dissimilar orientations and sizes but similar centers. Furthermore, the Bhattacharyya distance (9) is saturated in the range $[10^{-3}, 3]$. The lower boundary is due to the inverse of the distance in the loss function (10), and the upper boundary is because the loss can always be reduced by pushing the clusters further away from each other. This results in the clusters converging to exactly the set upper boundary value.

Second, a balanced soft assignment of the latent space data points to the fuzzy rules is encouraged with a loss based on the Kullback–Leibler divergence, as proposed in [28]:

$$\mathcal{L}_B = \sum_{i}^{c} \frac{\sum_k \Psi_i(k)}{N} \log \frac{\sum_k \Psi_i(k)/N}{1/c} \tag{11}$$

This balanced assignment loss encourages all clusters to have a uniform probability of being assigned a data point. While this is not always desired [28], it is beneficial in our case as it encourages the latent space data to be in proximity (have a low distance $d_i^2(k)$) to all clusters instead of just one. However, since the number of latent space data samples is based on the batch size, it affects the quality of the deep clustering. The batch size should be much larger than the number of clusters or fuzzy rules.

III. Experimentation

A. Stock market multi-horizon time series prediction

The proposed methodology was applied to the multi-step forecast of the closing prices of the Standard & Poor's 500 (&P 500, ticker symbol GSPC) stock market index. Tracking the stock performance of the largest companies listed on U.S. stock exchanges, the S&P 500 is regarded as a reliable indicator of the overall market. We examined multivariate data from other related market indicators: the VIX Volatility Index (VIX), a commonly referenced measure of the stock market's expectation of volatility based on S&P 500 index options, often referred to as the "fear index" or "fear gauge"; the Gold commodity price (GC=F); and the 5-year U.S. Treasury Yield (FVX), which tracks the performance of U.S. dollar-denominated domestic sovereign debt. The data were gathered from January 1st, 2001, to January 1st, 2023, then normalized using Min-Max scaling and split into training (80%), validation (10%), and testing (10%) sets. The data was not shuffled to preserve order, allowing for an examination of the trained model's performance on states not present in the training data. Since the stock market generally trends upward over time, the range of amplitudes of the test dataset differs significantly from the training dataset. The data for the main series (S&P 500) falls within the ranges [0, 0.533], [0.379, 0.705], and [0.629, 1] for the training, validation, and testing sets, respectively.

We compared our neuro-fuzzy network Evolver with the classic Autoregressive Integrated Moving Average (ARIMA) model and the Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) [2] for time series forecasting. The ARIMA metaparameters were selected with the Auto ARIMA method[3], which resulted in ARIMA(p,d,q), where p=4 is the number of autoregressive terms, d=1 is the order of differentiation for nonseasonal stationarity, and q=1 is the number of lagged forecast errors. The ARIMA order p=30 was also examined by initializing the statespace model with approximate diffuse values of 10^6 in case the variance becomes infinitely large during decomposition. The approach of training, validation, and testing is not applicable for the ARIMA model. For all forecasts, the univariate input sequence with N delayed time steps was used to fit the ARIMA model and then to forecast the H future time steps. The parameters of the ARIMA model are estimated for each input signal individually for all datasets. The following parameters were used for the LSTM model: hidden dimensions were 32 and 128, and the number of layers was 3. A linear layer was also used to transform the output into $\mathbb{R}^{N\times 1}$, and a number of time steps equal to the number of forecast horizons were taken as the output of the model.

A new LSTM and Evolver model was trained for each N/H combination for 200 epochs with the Adam Optimizer with a learning rate of 10^{-3} and $\beta = (0.9, 0.999)$, and a dropout rate of 10%. The methods were evaluated based on the Root Mean Squared Error (RMSE) measure, defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{M}\sum_{k=1}^{M}\left(\hat{\underline{y}}_k - \underline{y}_k\right)^2} \tag{12}$$

where $\hat{\underline{y}}_k$ is the future forecast at time step k, and M is the size of the dataset.

The meta-parameters of the Evolver network were selected experimentally as: 2 LSTM layers, 2 MHA layers, hidden dimension D_h=128, latent space dimension D_Z=2, number of attention heads h=4, order of auto-regression p=30 and p=4, differentiating order d=1, exogenous input order q=1, number of fuzzy rules $C = 16$. The batch size was $B = 313$ in all cases.

The results of the study are presented in Table I. The figures show the interpretable layers of the Evolver network for a test data sequence near the maximum of the time series. The heatmaps of the 2 MHA layers are shown in Figure 2, and the multivariate Gaussian clusters are shown in Figure 3. The consequent local ARIX models of the fuzzy layer are shown in Figure 4, and the combined output forecast in Figure 5.

The results of the case study, presented in Table I, demonstrate that the Evolver model is less prone to overfitting than the LSTM model when tested on the given data. The Evolver model provides more conservative forecasts, producing nearly linear slopes that mainly describe the future trend. In contrast, the LSTM network exhibits higher variance in predictions and, as a consequence, much larger prediction errors on the test data. Though the LSTM network performs well on the training data, the quality of its forecasts declines sharply on the test dataset, an indication of overfitting. This susceptibility to the amplitude ranges in the datasets, as set up in the experiments, likely contributes to this issue. Interestingly, more layers in the case of the LSTM network do not reduce the forecast error. Conversely, the Evolver model does not suffer from this problem, thanks to the use of simple ARIX local models, but it requires more epochs to train effectively due to the attention layer and individual recursive computation for each rule, making training slower. From the second column in Table I, it's apparent that extending the look-back period to N=150 yields no benefits for either the LSTM or Evolver networks.

The ARIMA models with $p = 4$ generally forecast a constant value without a trend, except when a pronounced trend is present over the entire look-back window. This approach results in forecasts with low RMSE, as the error is never too substantial. In comparison, the Evolver model produces cautiously optimistic predictions, revealing trends but avoiding the extremes observed with ARIMA, even when a clear trajectory exists in the look-back window. This behavior may be attributable to the training process, where continuing a downward trend leads to large RMSE errors if the series reverses direction (bounces).

B. Ablation Study

To examine the contribution of each layer on model accuracy, we conducted an ablation study by removing the main

[2]https://github.com/mihaozbot/Evolving-transformer

[3]https://github.com/robjhyndman/forecast/blob/master/R/arima.R

Table I
COMPARISON OF THE PROPOSED EVOLVER NEURO-FUZZY SYSTEM WITH CLASSIC MULTI-HORIZON TIME SERIES FORECASTING METHODS

N/H	60/30			150/30			150/60		
Dataset	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
Auto ARIMA(p=4, d=1, q=1)	0.0096	**0.0288**	0.0313	0.0091	**0.0290**	**0.0320**	0.0123	**0.0424**	**0.0441**
ARIMA(p=30, d=1, q=1)	0.0091	0.0303	**0.0311**	0.0100	0.0326	0.0355	0.0133	0.0472	0.0462
LSTM(D_h=256, 3 layers)	0.0124	**0.0407**	**0.1436**	0.0128	**0.0416**	**0.1536**	0.0163	0.0492	**0.1683**
LSTM(D_h=256, 1 layer)	0.0121	0.0386	0.0797	0.0118	0.0395	0.0981	0.0162	**0.0549**	0.1181
Evolver (Our, p=2)	0.0100	0.0303	0.0324	0.0098	0.0308	**0.0333**	0.0128	**0.0428**	0.0452
Evolver (Our, p=4)	0.0107	0.0320	0.0369	0.0098	0.0314	0.0338	0.0183	0.0450	0.0462
Evolver (Our, p=30)	0.0095	**0.0300**	**0.0321**	0.0094	**0.0301**	0.0336	0.0130	0.0455	**0.0442**

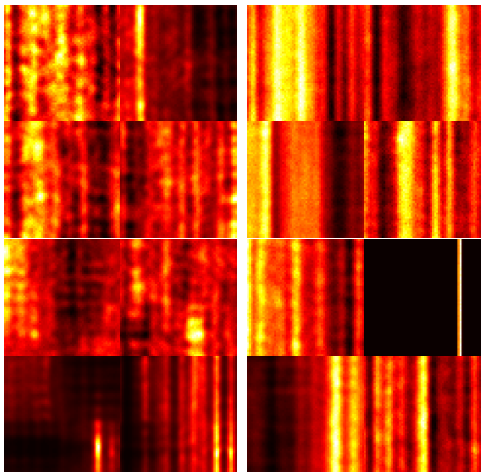

Figure 2. Heatmaps of each head of the temporal multi-head self-attention layer. The first layer (left) gives higher attention to details, while the second layer (right) gives attention to broader parts of the LSTM output sequence. The heatmaps are from the Evolver p=30 (top) and p=2 (bottom) and the test dataset with settings N=60 and H=30.

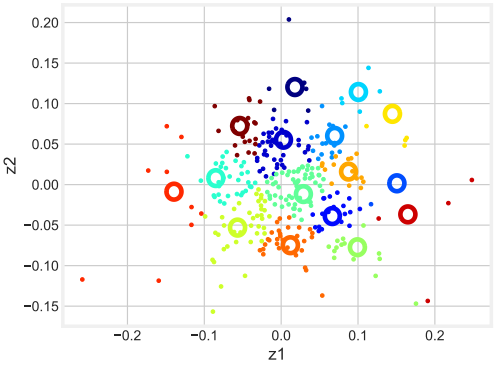

Figure 3. The neural network latent space $\underline{Z} \in \mathbb{R}^2$ with the fuzzy model multivariate Gaussian clusters and corresponding highest activation of membership functions for one batch of data with 313 samples. The membership functions of the fuzzy rules provide a label to the latent space data samples.

layers of the network. Specifically, we substituted the LSTM layer with a linear input embedding, removed the MHA layer, eliminated the cosine positional embedding, and set the (non-)exogenous inputs to zeros. The meta-parameters used were the same as for the $p = 30$ in the case study. The results of

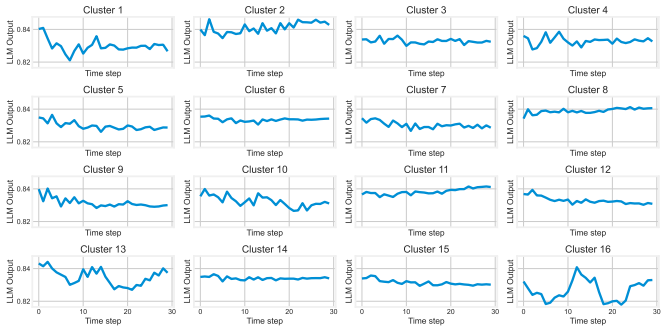

Figure 4. The forecasts of the local ARIX models p=30 of the fuzzy layer for one input sequence with N=60 and H=30. The final output is computed as a fuzzy combination of all local models.

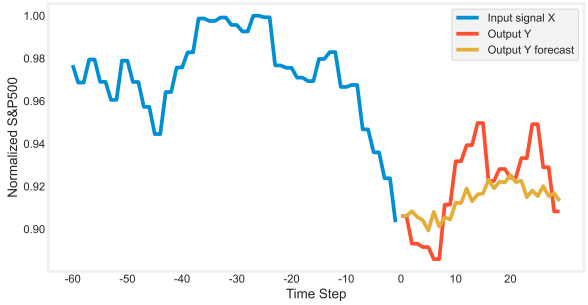

Figure 5. An example of the multi-horizon forecast of the Evolver $p = 30$ with the test data and N=60, H=30. This example is interesting because the model predicted a stock market bounce after a significant value drop.

this study are presented in Table II.

The removal of layers did not have a significantly large impact on the RMSE of the forecast, indicating that the main descriptive capability is attributable to the output fuzzy layer. This suggests that there are enough features such that the lack of some layers does not substantially degrade performance, and the model's complexity could be further reduced. However, this observation doesn't provide a full picture, as the neural network layers are still needed to create the latent space data for clustering and to enable the use of multivariate data. Further, the removal of layers affects the interpretability of the model.

The removal of the LSTM layer has the greatest increase of forecast error, however not significant. The positional

Table II
ABLATION STUDY

Removed element	Train	Valid	Test
Baseline (nothing removed)	0.0095	0.0300	0.0321
LSTM	0.0095	**0.0307**	0.0320
Temporal MHA	**0.0094**	0.0302	0.0321
Positional embedding	0.0095	0.0304	0.0320
(Non-)exogenous input	0.0095	**0.0299**	**0.0318**

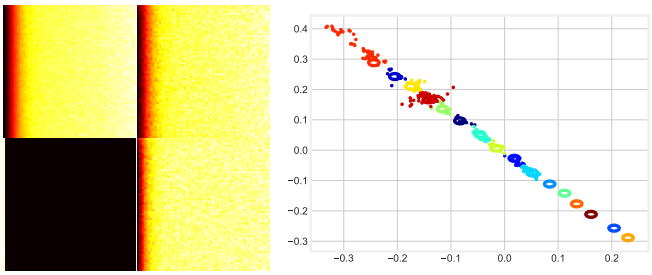

Figure 6. Ablation study interpretability reduction. The attention heatmap of the second layer without the cosine positional embedding (left) and the latent space samples and clusters without the MHA layer (right) on the test data.

embedding greatly influences the appearance of the attention heatmaps and, thus, the interpretability of the layer; see Figure 6. In this case, the attention is spread uniformly across the input sequence, with some heads giving more or less attention to the beginning of the sequence. Removing the attention layer impairs the interpretability of the connections between time steps and significantly affects the distribution of samples in the latent space. When the MHA is removed, the latent space samples are distributed along a single principal component. This may be due to the locality of the LSTM output, meaning that input sequences close in the time domain transform into similar latent space variables. The MHA layer, in contrast, transforms the input data with a higher degree of nonlinearity. Note that the latent samples in Figure 6 are skewed to the top left, and the lower right clusters have no samples in the test dataset, though they did in the training dataset. This suggests a time locality of the latent space samples where the leftmost samples correspond to the last time steps in the test dataset.

IV. DISCUSSION

The ARIMA model of higher orders may have some decomposition singularity issues during identification. However, our proposed ARIX model, which uses a similar autoregressive structure and is trained with the Adam optimization method, can use much higher orders without difficulty. Training an autoregressive model with a gradient descent method is more stable for higher system orders. Still, it's worth noting that the trained parameters might be biased in the classical system identification sense. The use of the neural network encoder allows the fuzzy system head to use multivariate data, as AR/ARX/ARIX/ARIMAX type models can handle only univariate data. The fuzzy model combines multiple local models in a manner similar to how samples are distributed and multiple models in a SARIMAX model, but with fuzzy model membership based on antecedent clusters.

The interpretability of the proposed method stems mainly from two aspects of the model: 1) the attention weights of the spatial attention, which show the importance of each input feature, while the weights of the temporal attention emphasize important time steps and reveal commonly related time steps; 2) the fuzzy system that generates the system's output, offering model transparency through the connection between the clusters and Local Linear Models (LLMs) [11]. The ARX type models are simple recursive models that can be examined for their time constants, trends, and patterns. However, interpretability requires the number of rules in the fuzzy system to be low, usually limited to fewer than 30. The fuzzy membership functions operate similarly to an attention mechanism, but instead of keys and queries, they use distance to clusters. This makes the mechanism interpretable for low-dimensional data (usually up to 3), but it has lower descriptive power and necessitates specialized clustering techniques.

The transparent structure of the model also aids in the debugging of the network. Some examples include the following. The attention layer and the clusters are very sensitive to the learning rate. The latent space samples and clusters might change rapidly between training epochs. The attention layer might show equal attention to all parts of the sequence, or multiple heads may focus on the same parts, indicating that the lower layers are inadequate. The latent space data might differ significantly during training and testing, signaling overfitting. The corresponding local models must exhibit variety in a healthy model.

The multi-head attention mechanism is slower to train compared to other layers due to a larger number of parameters. A high learning rate during training may cause the attention mechanism to focus on the first or final element of the LSTM output. While this is not conducive to the interpretability of the model, as it is not very informative, a lower learning rate produces better results but increases training costs. The primary information bottleneck of the network is the cluster membership functions that determine the activation. The learning rate must be high enough to enable the latent space data to change from one rule to another. Slow changes result in the latent space data remaining stuck with the same rule, unable to explore other configurations. Finding a compromise between these two problems requires further research and falls beyond the scope of this study.

V. CONCLUSION

In this study, we examined how a recurrent neural network architecture with multi-head self-attention can condense multivariate stock market data into features that can be utilized for an interpretable fuzzy multi-horizon time series forecast. The method enables the incorporation of multivariate data, thereby enhancing the information available as input to the fuzzy inference systems, which typically have lower descriptive power and encounter difficulties with multivariate

data. The proposed approach demonstrates comparable performance with established ARIMA and LSTM networks that are commonly used for multi-horizon time series forecasting while offering valuable insights into the model structure and information flow through the network. The temporal attention mechanism highlights the significance of delayed time steps, and the fuzzy inference system offers interpretability with the cluster-based membership functions and simple consequent models. Nonetheless, there remain opportunities for incorporating further interpretable mechanisms into the proposed fuzzy system that were not implemented in this study [11]. The initialization of clusters could also be enhanced with an initial clustering method, and the dependency of clustering on the batch number is a limitation to be addressed. These initial results are promising, and with continued research, they could contribute to a deeper understanding of stock market behavior.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," vol. 2017-December, 2017.

[2] C. Xie, D. Rajan, and Q. Chai, "An interpretable neural fuzzy hammerstein-wiener network for stock price prediction," *Information Sciences*, vol. 577, pp. 324–335, 10 2021.

[3] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, pp. 1748–1764, 10 2021.

[4] S. Chang, "Deep clustering with fusion autoencoder," no. arXiv:2201.04727, Jan 2022, arXiv:2201.04727 [cs]. [Online]. Available: http://arxiv.org/abs/2201.04727

[5] M. Ozbot, E. Lughofer, and I. Skrjanc, "Evolving neuro-fuzzy systems based design of experiments in process identification," *IEEE Transactions on Fuzzy Systems*, p. 1–11, 2022.

[6] E. Lughofer and M. Sayed-Mouchaweh, "Autonomous data stream clustering implementing split-and-merge concepts – towards a plug-and-play approach," *Information Sciences*, vol. 304, pp. 54–79, 5 2015.

[7] Igor Škrjanc and Jose Iglesias and Araceli Sanchis and Daniel Leite and Edwin Lughofer and Fernando Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey," *Information Sciences*, vol. 490, pp. 344–368, 7 2019.

[8] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "Panfis: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, pp. 55–68, 1 2014.

[9] C. Garcia, A. Esmin, D. Leite, and I. Škrjanc, "Evolvable fuzzy systems from data streams with missing values: With application to temporal pattern recognition and cryptocurrency prediction," *Pattern Recognition Letters*, vol. 128, pp. 278–282, 12 2019.

[10] S. Jahandari, A. Kalhor, and B. N. Araabi, "Online forecasting of synchronous time series based on evolving linear models," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, pp. 1865–1876, 5 2020.

[11] E. Lughofer, "On-line assurance of interpretability criteria in evolving fuzzy systems – achievements, new concepts and open issues," *Information Sciences*, vol. 251, pp. 22–46, 12 2013.

[12] P. Angelov, "Evolving takagi-sugeno fuzzy systems from streaming data (ets+)," *Evolving Intelligent Systems: Methodology and Applications*, pp. 21–50, 4 2010.

[13] E. Lughofer, J. L. Bouchot, and A. Shaker, "On-line elimination of local redundancies in evolving fuzzy systems," *Evolving Systems*, vol. 2, pp. 165–187, 9 2011. [Online]. Available: https://link.springer.com/article/10.1007/s12530-011-9032-3

[14] D. Dovžan, V. Logar, and I. Škrjanc, "Implementation of an evolving fuzzy model (efumo) in a monitoring system for a waste-water treatment process," *IEEE Transactions on Fuzzy Systems*, vol. 23, pp. 1761–1776, 10 2015.

[15] I. Skrjanc, "Cluster-volume-based merging approach for incrementally evolving fuzzy gaussian clustering-egauss+," *IEEE Transactions on Fuzzy Systems*, vol. 28, pp. 2222–2231, 9 2020.

[16] W. Guo, K. Lin, and W. Ye, "Deep embedded k-means clustering," no. arXiv:2109.15149, Sep 2021, arXiv:2109.15149 [cs]. [Online]. Available: http://arxiv.org/abs/2109.15149

[17] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," 1 2018. [Online]. Available: https://arxiv.org/abs/1801.07648v2

[18] S. Aryal, D. Nadarajah, P. L. Rupasinghe, C. Jayawardena, and D. Kasthurirathna, "Comparative analysis of deep learning models for multi-step prediction of financial time series," *Journal of Computer Science*, vol. 16, no. 10, p. 1401–1416, Oct 2020.

[19] A. Yadav, C. K. Jha, and A. Sharan, "Optimizing lstm for time series prediction in indian stock market," *Procedia Computer Science*, vol. 167, pp. 2091–2100, 1 2020.

[20] T. Guo, T. Lin, and N. Antulov-Fantulin, "Exploring interpretable lstm neural networks over multi-variable data," no. arXiv:1905.12034, May 2019, arXiv:1905.12034 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1905.12034

[21] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" no. arXiv:2205.13504, Aug 2022, arXiv:2205.13504 [cs]. [Online]. Available: http://arxiv.org/abs/2205.13504

[22] P. Ghosh, A. Neufeld, and J. K. Sahoo, "Forecasting directional movements of stock prices for intraday trading using lstm and random forests," *Finance Research Letters*, vol. 46, p. 102280, 5 2022.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, p. 1735–1780, Nov 1997.

[24] G. Prasad, E. Swidenbank, and B. W. Hogg, "A local model networks based multivariable long-range predictive control strategy for thermal power plants," *Automatica*, vol. 34, no. 10, p. 1185–1204, Oct 1998.

[25] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," no. arXiv:1402.1128, Feb 2014, arXiv:1402.1128 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1402.1128

[26] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "Multi-head attention: Collaborate instead of concatenate," no. arXiv:2006.16362, May 2021, arXiv:2006.16362 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2006.16362

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

[28] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," Jan 2018. [Online]. Available: https://arxiv.org/abs/1801.07648v2