Hybrid ML: Implementation of a Mixture of Experts (MoE) ensemble algorithm in a soil moisture forecasting task.

1st Murcia Eduart A.
University of Florida.
Gainesville, Florida
emurciabotache@ufl.edu

I. Abstract

The promise of Machine Learning (ML) and deep learning (DL) algorithms to become the state-of-the-art in time series (TS) forecasting tasks, as in many other applications, has yet to be entirely delivered. DL algorithms specially designed to handle sequential data and the temporal dependence of the data explicitly have been incapable of dealing with specific properties of the structure of the TS. High dimensionality, changes in the data structure (drifts), and mixtures of long and short-term dependencies, among others, are part of the list of limiting TS features. Hybrid approaches have been proposed to overcome the limitations that individual approaches often need help to handle. Nevertheless, several approaches can be taken to combine the power of multiple algorithms. A straightforward hybrid way that promotes collaboration is just averaging the results from multiple algorithms. However, despite the well-proven benefits of this approach over non-hybrid ones, it has also been found that overfitting issues could limit the capacity of this approach. An alternative way to proceed is to use what has been named a Mixture of Experts (MoE). MoE approaches promote specialization by building multiple 'local' models contributing to the global model. In this research, I aim to implement and evaluate the performance of an MoE forecasting architecture in a soil moisture forecasting task. For this, I used 3 ML-DL 'expert' forecasting algorithms and a Gaussian Mixture model acting as a gating network to output a probability distribution over the expert algorithms that determine the contribution of each expert to the final prediction. The input data comprises three sets of three years of soil moisture data collected every 30 minutes. The MoE model was trained through an Expectation-Maximization (EM) approach. Our results show that the convergence of the algorithm toward minimum loss functions was not guaranteed and therefore the performance of the models was highly variable between multiple trials. The model was capable only of capturing general tendencies but there were sections of the forecasted data that were highly erroneous. We conclude that further strategies would be necessary to be applied to improve the capacity of the model to produce the expected results.

II. Introduction

A. Context

Soil moisture forecasts are essential for supporting effective crop management and irrigation scheduling processes in modern agriculture, particularly given the challenging environmental and policy-related conditions that farmers face today (Brinkhoff et al., 2019). By providing farmers with accurate information about this variable several days in advance, they can make more informed decisions regarding irrigation schedules, nutrient management, and planting times.

Modern agricultural systems have a wide range of sensors that can store and transfer large volumes of data in real and near real-time, providing detailed information on the system's high spatial and temporal variability. This data can be used for modern data-driven forecasting algorithms to provide site-specific informed predictions that can improve the crop management and irrigation scheduling process. Nevertheless, processing such vast amounts of data and extracting meaningful information can be a challenging task (Torres et al., 2020). In particular, time series (TS) data analysis presents several limiting obstacles, such as high dimensionality, temporal dependence, non-linearity, and non-stationarity, which can compromise accurate forecasting and decision-making (Cerqueira et al., 2020).

Machine learning and deep learning algorithms have shown significant potential in uncovering complex patterns in large datasets. However, research indicates that the performance of these algorithms can still be limited (Makridakis et al., 2018). To overcome these limitations, researchers have proposed hybrid approaches incorporating multiple algorithms in an ensemble framework and leveraging modern signal preprocessing techniques (Xu et al., 2021), (Jin et al., 2020), (He et al., 2020). The development of hybrid algorithms has been directed toward two approaches: 1.) combining time series signal preprocessing techniques with modern ML algorithms, and 2.) the integration of an ensemble of

algorithms.

In my research, I am first exploring both approaches individually to understand the capacities of each of them better to improve forecasting accuracy, and then, I intend to implement a combination of both. The first approach (i.e., a combination of ML algorithms with signal decomposition techniques such as singular spectrum analysis and empirical mode decomposition) was already completed, and it was found a beneficial effect of using signal decomposition combined with a Long Short Term Memory with autoencoder (LSTM-ED) and random forest (RF). In this second part of my research, I am exploring the capacity of a Mixture of Experts (MoE) algorithm that promotes specialization on parts of the data by developing local experts to improve forecasting accuracy. For this, I developed an algorithm that uses three expert forecasters to produce individual predictions, each specialized in a certain region of the input space. Then the outputs are combined using a Gaussian Mixture gating network. The final optimization process of this MoE algorithm is completed using an expectation-maximization (EM) algorithm in which, through several iterations, the responsibilities of each expert for each data point are computed in the E-step. Then the parameters of each expert and the gating network are updated in the M-step.

III. Description

The development of this hybrid ML forecasting approach includes the following steps

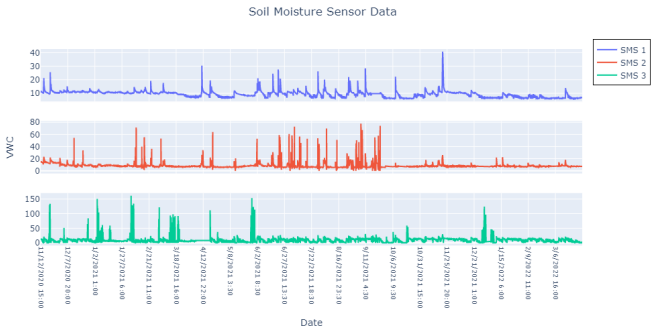

Fig. 1: Raw soil moisture time series. The three datasets used are presented.

1) We conducted an Exploratory Data Analysis (EDA) on the soil moisture and soil temperature data to assess their quality, identify missing values and anomalies, and understand their temporal distribution, trends, seasonal components, and complexity. This step included common time series quality control approaches such as consistency tests, visual inspection, missing data assessment, and imputation. Also, we implemented similarity analysis and causality tests better to understand the patterns and structure of the data. The results of this step are not fully presented in this document as it is not the main focus of this stage of the research but was a crucial part of the correct implementation of all the procedures.
2) Next, we explored the first stage of our hybrid system, namely, implementing two signal decomposition techniques with an LSTM-ED and RF to reduce the complexity of the raw data. In this stage, we implemented SSA and EMD to reduce the complexity of the data, and then we implemented an LSTM-ED and an RF algorithm for each component. The results of this stage suggest that using EMD can increase the performance of the forecasting algorithms; nevertheless, it still requires further implementation of other techniques, mainly to deal with the drift (extreme and sudden changes in the data structure). For this reason, we decided to implement an MoE ensemble algorithm.
3) The implementation of the MoE model uses different types of expert forecasters to generate multiple predictions of the training data. Then, a Mixture of Gaussian gating networks takes the outputs of the expert models and uses a Gaussian distribution per component to determine the weights (i.e., the probabilities of each expert being appropriate for the given part of the input space) assigned to each expert for the given input. The implementation of the gating network contains a final dense layer that is in charge of providing the logits (unnormalized log probabilities) and the parameters of the Gaussian component. First, the logits are used as inputs for the 'tfp.distribution.Categorical' function to model the categorical distribution of the experts. Then, the distribution of the Gaussian components is modeled using the 'tfp.distribution.Normal' function, whose parameters are the components' mean and the standard deviation. Finally, using an EM algorithm, the weight of the experts is computed (E-step), and the parameters of the experts and gating network are updated in the M-step.
4) Before proceeding with the implementation of a hyperparameter tuning and training routine, I quick-explored several algorithms as experts (including simple MLP, Autoencoders, Convolutional Neural Networks (CNN), recurrent neural networks (RNN) such as LSTM and GRUs, attention-based RNNs. From there, I selected the final algorithms: an Autoencoder with multiple attention mechanism LSTM network, a Multi-head (transformer) network, and a GRU with an attention mechanism network. In addition, I also incorporated some tools for improving the convergence and overall performance of the algorithms, such as schedulers, dropout layers, multiple initializers, and early stopping. Finally, during this stage, I also implemented a routine using the negative log-likelihood (NNL) loss function to directly optimize the data distribution rather than minimizing the mse, as it is a common strategy when working with time series data. However, I did not observe significant benefits from using this optimization strategy.

5) The hyper-parameter tuning optimization process was performed using Optuna, a hyper-parameter optimization framework known for efficiently searching large spaces and pruning unpromising trials. For this, the selected hyper-parameters were: the learning rate, the dropping rate, the kernel initializers, and the activation function. The training set consisted of the first 10000 data points. Because this research is dealing with time series, a walk-forward validation strategy was implemented, instead of regular cross-validation, with a window size of 500-time steps and a step size of 50. Due to computational cost and processing time, we implemented ten iterations, acknowledging that in real-life scenarios, this number should be increased—the selected hyper-parameters were then used to train the algorithm and then used on the test sets. The mean squared error (mse) was used as a loss function, and the mean absolute error (mae) as additional metric performance in both training and testing sets.
6) The work presented in this document does not use the pre-trained models from the previous stage; instead, it uses the EM algorithm (with ten iterations) to train and optimize the parameters of the experts. Due to the complex nature of the time series data, this approach could be limiting as the computational cost of training experts and the gating model can limit the extent to which the experts are trained. Therefore, the next step of this research will consist of updating the implementation of the MoE such that it receives pre-trained models (i.e., with routines specialized in optimizing the algorithms) and focuses on optimizing the gating network.
7) After tuning the hyperparameters, we trained the model in the entire training set (10000-time steps), saved the model, and, in a separate notebook, loaded the model and evaluated the test set's performance.
8) To increase the statistical significance of my results, I repeated the entire process for three different datasets (i.e., data collected from three different datasets).

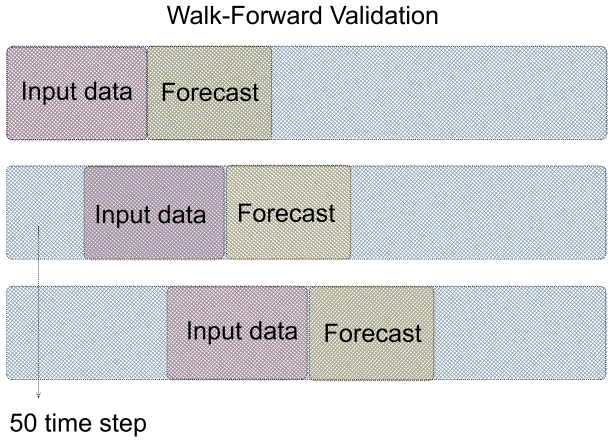

Fig. 2: Walk forward validation strategy

IV. Related work

Hybrid ML modeling approaches have been proposed and implemented for several time series and forecasting studies and competitions such as the Makriadakis (M-3, M-4, and M-5) competitions (Makriadakis et al., 2018). They have proved in a wide range and representative datasets the capabilities of hybrid approaches to increase the model performance and interpretability of the algorithms. The series of hybrid approaches includes using data pre-processing techniques and signal decomposition techniques, and combining multiple algorithms in innovative ways to boost the performance of individual models.

MoE methods are one of the ensemble approaches, for ML, that has called the attention because of their capabilities to combine and increase the performance of individual ML algorithms. Their applications range from forecasting task to classification and clustering tasks. They are based in the principle of divide and conquer, and they work by creating local experts for the input space that contribute to the global solution (Armi and Zarepour., 2021). They are more than simple averaging and can provide better generalization capabilities that them.

Han et al. (2022) introduced a mixture of heterogeneous experts framework named MECATS that is claimed to be capable of providing parallel probabilistic forecasts using aggregation hierarchy. The framework claims to be 'distribution-free' and independent of the selection of algorithms.

Wu et al. (2021) presented a dynamic Gaussian mixture deep generative forecasting approach for sparse time series that uses a gating mechanism for dynamically tuning the GMM. The authors experimented on multiple real-life datasets and showed the potential benefits of this approach.

V. Experiments and evaluation.

A. Building the MoE architecture: exploring multiple expert algorithms and strategies to improve convergence bias reduction, and generalization capabilities.

The first step after doing preprocessing of the data was to build the architecture of the MoE model. For this, we build a function whose pseudo code is provided below.

- Initialize an empty list of experts of size three.
- Loops on a routine to create the architecture of three experts and stores them in the initialized list.
- Initialize the gating network.
- Creates the feed-forward architecture of the gating network.
- Creates the MoE model with an input layer.
- Calls the experts.
- Calls the gating models.
- Computes the weighted outputs by performing a matrix multiplication of the gating output and the expert outputs.

- Sum the weighted outputs.
- Returns the MoE, the experts and the gating model.

Next comes the EM algorithm to train the experts and the gating network.

- For this, an iterative process, whose breaking point is either the number of iterations or a learning rate smaller than 1e-06, is started.
- In the E-step the computation of responsibilities of each expert for each data point is performed. This includes a computation and a normalization of the gating output
- In the M-step for each expert model the expert outputs and loss functions are computed, then the gradients are calculated and then used for updating the expert model variables
- Next, the learning rate is updated with the scheduler
- The gating output and loss functions are computed
- The gating gradients are computed
- The gating gradients are computed and then used for updating the gating parameters.
- These steps are repeated until one of the breaking conditions is met.

Once we built the MoE architecture we proceed to experiment with multiple expert models. We started with three shallow and fully connected neural networks whose performance was not the best. We tried to improve the performance (i.e., convergence, bias reduction, and generalization capabilities) by including additional layers, adding dropout layers (performing quick-experiments with different drop rates), and using alternative kernel initializers such as He_Normal. In addition, we tried using clipping gradients in the gating network, varying the learning rate, adding schedulers and early stopping.

The next step was to include a encoder layer to compress the representation of the input data in a lower dimensional space. We expected this provide an improvement on the performance considering the complexity of the input data. Similar strategies to the ones described before were used in this expert model. However, we did not observe a significant improvement in the performance.

The next step was to included a CNN as one of the experts. By including this algorithm as an expert we expected to have additional capacity to extract temporal patterns from the input data, model more complex relationships between input data and target variables (i.e., between past soil moisture and temperature data and future soil moisture data). However, as in the previous cases we did not find a significant improvement on the performance when using this algorithm.

The next step was to include an LSTM neural network with an attention mechanism. This architecture is well know for its capacity of learning and remembering long-term dependencies, therefore, being suited for long sequences of time. This feature, together with its capacity for model complex relationships between the input and output data provided results that suggested this would be a good model to include in the list of experts. In addition, this algorithm provides a feature that we did not explore yet but that will be useful in future implementations, this is its capacity to handle variable length-input sequences.

Finally, we implemented two algorithms that provided better results than the previous ones and were selected to complete the list of three experts for the MoE. The first one is a Transformer-based attention mechanism that due to the usage of a multi-head attention mechanism allows the model to learn multiple important features of the input data simultaneously. This feature is commonly useful in complex task such as forecasting. The second algorithm is a combination of GRU layers with an attention mechanism. This algorithm has similar features and benefits than the LSTM.

The list of tools used with the purpose of improving the model convergence, bias reduction and generalization capabilities of the model include:

- Using multiple Kernel initializers to improve the convergence of the algorithms.
- Dropout layers with multiple drop rates to improve the generalization capabilities.
- Trying multiple number of units per layer and adding layers to the architectures to improve bias reduction.
- Adding early stopping and learning rate schedulers.
- Using clipping gradients on the gating network
- Trying a direct optimization of the data distribution, instead of using mse, by implementing an NNL loss function.
- Implementing a moderately large hyper-parameter tuning routine with some of the most relevant hyper-parameters such as the learning rate, dropout rate, kernel initializer, and activation function in the input and hidden layers.
- Using stochastic gradient descent (SGD) with momentum too help accelerate the convergence of the training process.

B. Hyper-parameter tuning, training final algorithm and performance report

The hyper-parameter tuning process was implemented using Optuna, which is an efficient optimization framework that allows faster and more exhaustive tuning process. For training the first 10000-time steps were used. Also, the routine includes a walk forward validation process with a window size of 500-time steps and a step size of 50. Given that each iteration includes the optimization of the experts and gating network through the EM algorithm, we had to limit the number of trials and the number hyper-parameters. We chose 5 trials and selected the most important hyper-parameter (i.e., the learning rate) and the dropout rate, kernel initializer, and activation function in the input and hidden layers. We

TABLE I: Optimized hyper-parameters and training loss value achieved

	Sensor 1	Sensor 2	Sensor 3
Learning rate	0.0080	0.0017	0.0089
Kernel Initializer	None	None	glorot normal
Activation function	ReLU	tanh	sigmoid
Dropout rate	0.337	0.114	0.229
Training loss	0.0047	0.0024	0.4985

acknowledge that real-life implementations of this algorithm might require adding additional trial to the hyper-parameter routine.

Table 1 depicts the final hyper-parameter values and the values of the mse loss function for each of the three sensors. From there we can observe that the model was capable of minimizing the loss (mse) function in all three cases. We also observe that the learning rate is in the order of X10ˆ3. the activation function was different each time and the kernel initializer was None.

We observed that adding the SGD optimization algorithm with momentum set to 0.9 helped to escape the local minimum in which the algorithm was trapped, however, it also undermined the stability of the algorithm. After running several time the training routine the convergence of the algorithm was not always guaranteed. We tried implementing additional optimizers such as the Nesterov Accelerated Gradient (NAG), nonetheless, the convergence of the loss function to a minimum was not guaranteed either. Table 2. shows the best (minimum) values achieved in the training routine. From there we can observe the low performance of the models on the final training stage. Here it is worth mentioning that the algorithm was set to forecast the entire dataset, however, in real life applications in agriculture shorter periods of time would be needed.

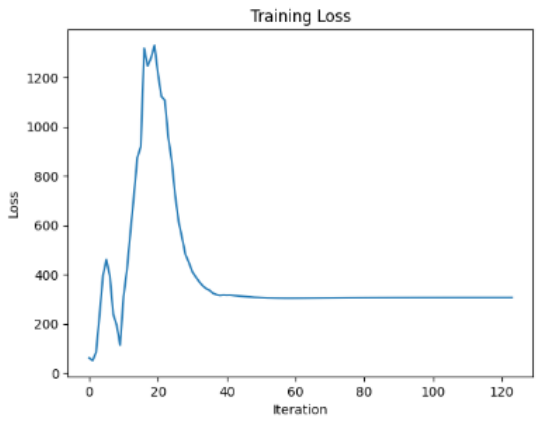

Fig. 3: Loss function in training stage of sensor 3

TABLE II: Training performance metrics

	Sensor 1	Sensor 2	Sensor 3
mse	43.871	81.542	75.453
mae	6.451	8.475	9.154

TABLE III: Performance metrics on the test set

	Sensor 1	Sensor 2	Sensor 3
mse	71.377	68.436	119.573
mae	7.742	7.461	7.388

C. Testing performance report

During the final training of the algorithm with the entire training dataset it was necessary to switch the optimizer from ADAM to stochastic gradient descent (SGD) to add momentum to the learning rate as it was getting caught in local minimum and the loss functions was not being minimized. After adding the SGD the convergence of the model improved significantly, however, the loss (mse) was still far from being ideal. Table 3 shows the mae and mse obtained with the final trained model. From there we can see that both metric performance were bad, especially the mse which shows the presence of extreme values (outliers) that were forecasted.

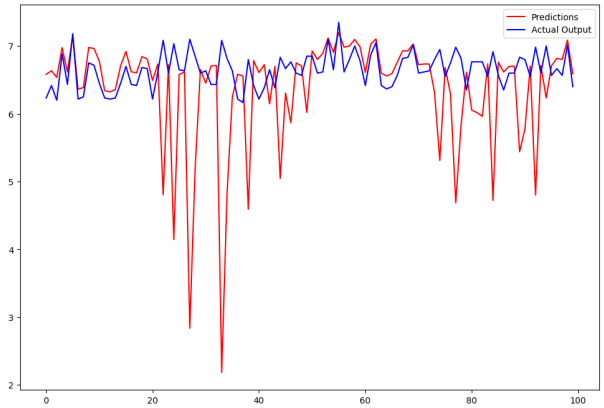

Fig. 4: Predicted (red) and actual values (blue) in the test set for sensor 1

VI. Limitations and Future Work

One of the main limitations of the current approach comes from the fact that our implementation trains both the experts and the gating network at the same time. Considering the complexity of the input data and computational costs of training the algorithms it could be a better approach to use pre-trained models and focus the optimization process of this stage on the gating network. Thus, adapting the function to receive the outputs of pre-trained models could result beneficial. In addition, for the purposes of this exercise we had to limit the optimization of the algorithms in some steps and also use time step of the sliding window (we used 50 instead of 1). This speeds up the process but limits the capacity of the model to learn the time-dependent underlying structure of the data.

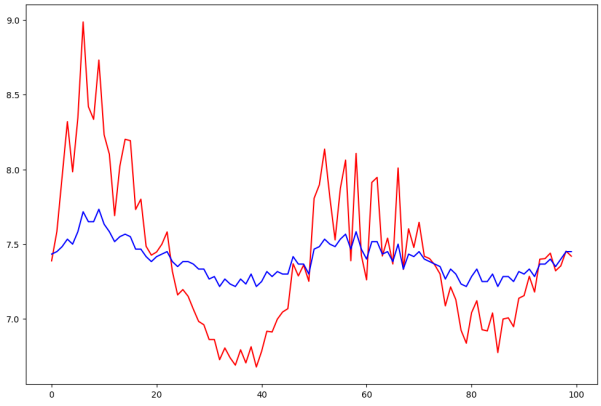

Fig. 5: Predicted (red) and actual values (blue) in the test set for sensor 2

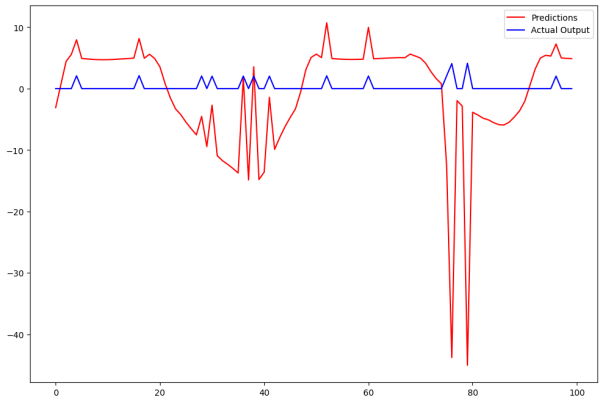

Fig. 6: Predicted (red) and actual values (blue) in the test set for sensor 3

VII. Summary and conclusions

We implemented a MoE algorithm that integrates the forecast outputs of three experts using a gating function that uses GMM to determine, point by point, the contribution of each model to the data. The parameters of the experts and gating function are optimized using and EM approach. During the E-step, the gating function is fixed and the responsibilities of each expert for each data point are computed. During the M-step, the experts are updated based on these responsibilities, and then the gating function is updated based on the updated experts. The three selected experts were an LSTM with an attention mechanism, and GRU with an attention mechanism, and a transformer-based attention mechanism. The results indicate that the MoE was only capable of capturing certain features of the input data. For the three datasets that were used to train and evaluate the performance of the MoE, we observed high instability on the convergence of the model during training, high mse and mae on the training and test sets, and the presence of forecasted values in the test stage that were out of range.

Thus, our initial hypothesis of an MoE with GMM being able to improve the forecasting accuracy of soil moisture forecasting was rejected, at least with architecture and experts that we used. However,some previous MoE implementations have been successfully implemented in forecasting tasks, therefore we believe modifications to the current architecture and implementation of the MoE could significantly improve the performance. Some of the suggested approaches to take in the future include using pre-trained models rather than training the models and the gating network sequentially through the EM algorithm, implementing different experts, extending longer hyper-parameter tuning optimization routines, change the architecture of the gating function towards one capable of capturing the complex relationships between input and output data.

VIII. References

Brinkhoff, J., Hornbuckle, J., and Ballester Lurbe, C. (2019). “Soil moisture forecasting for irrigation recommendation.” IFACPapersOnLine, Vol. 52, Elsevier B.V., 385–390 Ref

Cerqueira, V., Torgo, L., and Mozetič, I. (2020). Evaluating time series forecasting models: an empirical study on performance estimation methods, Vol. 109. Springer US Ref

Han X., Jing Hu, and Joydeep Ghosh. (2022). MECATS: Mixture-of-Experts for Probabilistic Forecasts of Aggregated Time Series. Ref

He, X., Luo, J., Li, P., Zuo, G., and Xie, J. (2020). “A hybrid model based on variational mode decomposition and gradient boosting regression tree for monthly runoff forecasting.” Water Resources Management, 34(2), 865–884. Ref

Xu J., Yang, N.-X., Wang, X.-Y., Bai, Y.-T., Su, T.- L., and Kong, J.-L. (2020). “Hybrid deep learning predictor for smart agriculture sensing based on empirical mode decomposition and gated recurrent unit group model.” Sensors, 20(5), 1334 Ref

Armi L, Abbasi E, and Jamal Zarepour-Ahmadabadi. (2021). Mixture of ELM based experts with trainable gating network.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). “Statistical and machine learning forecasting methods: Concerns and ways forward.” PloS one, 13(3), e0194889. Ref

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., and Troncoso, A. (2021). “Deep learning for time series forecasting: A survey.” Big Data, 9(1), 3–21 PMID: 33275484. Ref

Wu Y., Jingchao Ni, Wei Cheng, Bo Zong, Dongjin Song, Zhengzhang Chen, Yanchi Liu, Xuchao Zhang, Haifeng Chen, and Susan Davidson. (2021). Dynamic Gaussian Mixture based Deep Generative Model For Robust Forecasting on Sparse Multivariate Time Series.Ref

Xu, Z., Zhou, J., Mo, L., Jia, B., Yang, Y., Fang, W., and Qin, Z. (2021). “A novel runoff forecasting model based on the decompositionintegration-prediction framework.” Water, 13(23), 3390 Ref