Time Series Mini-Project

MVA 2025-2026

Time Series Forecasting Using LSTM Networks: A Symbolic Approach

S. Elsworth and S. Güttel (2020)

Paul Rongieras
paul.rongieras@ens-paris-saclay.fr

Valentin Smague
valentin.smague@ensae.fr

January 6, 2026

1 Introduction and contributions

Time series forecasting is a challenging task, even with the recent development of deep learning models and especially Recurrent Neural Networks (RNNs). When dealing with raw time series, there is currently no reliable model that systematically outperforms traditional statistical methods. Furthermore, there exist significant limitations to using neural networks for forecasting, such as their high sensitivity to hyperparameters and to the initialization of random weights, as well as the computation time required for training. In this context, S. Elsworth and S. Güttel propose a new approach called ABBA-LSTM, where time series are converted into a sequence of symbols through an Adaptive Brownian Bridge-based Aggregation (ABBA) symbolic representation, before using a Long Short-Term Memory (LSTM) neural network. The authors demonstrated on various datasets that this model is faster to build, train, and use for forecasting than a raw LSTM model, while achieving similar or better performance, according to metrics such as the sMAPE score.

Consequently, we decided to reimplement the entire proposed method on the original datasets as well as on new ones, to verify if similar results could be obtained. The sMAPE scores obtained were of a similar order of magnitude to those reported in the paper. We also observed that the training and forecasting runtimes of the ABBA-LSTM model were indeed faster than those of the raw LSTM model. Both Paul and Valentin implemented the two models but tested them on different datasets. After extensive hyperparameter tuning, Paul began drafting the report while Valentin focused on optimizing the model's performance. The final report was written equally.

Our contribution is as follows. Since the authors' code was outdated due to old dependencies and PyTorch versions, we completely redesigned the pipeline using 100% of our own code. We implemented various ABBA methods with different clustering techniques, parameters, and inverse transformations. Reproducing their experiments required adjusting hyperparameters, so we added a grid search and studied parameter influence. We also improved training by exploring hyperparameters such as the learning rate (that they fixed), and by introducing temporal cross-validation to avoid overfitting on the tail of the series. Finally, we implemented a stochastic symbol prediction strategy to prevent the model from always forecasting the most probable symbol.

2 Method

The approach implemented in this project, denoted as ABBA-LSTM, shifts the paradigm of time series forecasting from a regression task on raw numerical values to a classification task on symbolic representations. The method operates in a three-stage pipeline: (1) transforming the time series into a symbolic string using the ABBA framework, (2) training a Recurrent Neural Network (RNN) to predict future symbols, and (3) reconstructing the numerical time series from the predicted symbols.

2.1 Symbolic Representation (ABBA)

The ABBA method (Adaptive Brownian Bridge-based Aggregation) reduces the dimensionality of the time series $T = [t_1, t_2, \ldots, t_N]$ while preserving its essential shape features. It consists of two phases: compression and digitization.

Compression: The time series is approximated by a polygonal chain. An adaptive piecewise linear approximation partitions T into m segments. Each segment j is characterized by a tuple

$(len_j, inc_j) \in \mathbb{R}^2$, where len_j represents the length of the segment (time duration) and inc_j represents the increment in value (amplitude change). The compression is controlled by a tolerance parameter tol, which dictates the maximum reconstruction error allowed.

Digitization: The resulting sequence of tuples (len, inc) is then discretized. The tuples are grouped into k clusters using α clustering algorithm, where data points are grouped by connecting those whose distance is below a threshold α. We also added a parameter to control the maximum number of clusters, merging the closest clusters when this limit is reached. Each cluster is assigned a unique symbol from a finite alphabet $\mathbb{A} = \{a_1, \ldots, a_k\}$. The time series T is thus converted into a symbolic string $S = [s_1, s_2, \ldots, s_m]$ where each $s_i \in \mathbb{A}$.

Patched Reconstruction: A key contribution of the article, which we implemented, is the patching reconstruction technique. Instead of reconstructing the signal using straight lines (which yields a polygonal look), each symbol is associated with a patch. A patch is defined as the pointwise mean of all raw time series segments belonging to a specific cluster. During the inverse transformation, these smooth patches are stitched together, resulting in a forecast that faithfully mimics the local shapes and behaviors of the historical data.

2.2 LSTM Network Architecture

The core predictive model is a Long Short-Term Memory (LSTM) network. Unlike standard time series forecasting where the model outputs a continuous value $\hat{y} \in \mathbb{R}$, our model treats the problem as sequence generation. The network receives a sequence of one-hot encoded symbols. At each time step t, the LSTM cell maintains a hidden state $h^{(t)}$ and a cell state $c^{(t)}$ updated via the standard gating mechanisms described in Appendix.

Classification Head: The final layer is a dense layer with k neurons (corresponding to the alphabet size) followed by a Softmax activation function. This outputs a probability distribution over the alphabet $\mathbb{A}$. The model is trained to minimize the Categorical Cross-Entropy loss, contrasting with the Mean Squared Error (MSE) typically used in raw time series regression.

2.3 Training and Forecasting Strategy

For training, we split our dataset into training and test sets using an 80/20 ratio. The test set corresponds to the final portion of the time series and is used to evaluate the forecasting performance on future data. The training set is further divided into multiple validation sets using temporal cross-validation. These splits are performed at different points in time to avoid validating only on the end of the training series, which could lead to biased or misleading results. We tune several hyperparameters, including the learning rate, number of epochs, and early-stopping patience. Dropout is also applied to reduce overfitting, particularly when training on short sequences.

For forecasting, we employ an iterative strategy. The model predicts the next symbol $\hat{s}_{m+1}$, which is then fed back into the network to predict $\hat{s}_{m+2}$, and so on. Finally, the generated string of symbols is converted back to numerical values using the patching inverse-transform described previously, and we use a linear interpolation to have the number of forecasted points needed.

3 Data

To assess the robustness and versatility of the ABBA-LSTM framework, we conducted experiments on a diverse set of time series ranging from deterministic synthetic signals to complex real-world datasets from the UCR Time Series Classification Archive.

Synthetic Verification: As a proof of concept, we first generated a synthetic sine wave defined by $y(t) = \sin(t)$ over the interval $t \in [0, 20]$ ($N = 200$). This dataset is perfectly periodic, smooth, and deterministic. It serves as a "sanity check" to verify that the ABBA compression accurately captures simple geometric shapes and that the LSTM can learn trivial periodic patterns without overfitting or collapsing.

The "HouseTwenty" Challenge: We included the "HouseTwenty" dataset to highlight a specific limitation of standard regression models. This series is characterized by sharp switching between two distinct value intervals: $[340, 370]$ and $[2450, 2550]$. Data diagnosis shows a clear bi-modal distribution. A standard MSE-based regression model (such as a raw LSTM) typically minimizes error by predicting the mean of these two states (≈ 1400), a value that is physically impossible in the context of the data. In contrast, the symbolic nature of ABBA allows the model to digitize these levels into specific symbols, effectively capturing the switching logic without attempting to average the distinct states.

Real-World Benchmarks (UCR): We selected two datasets from the UCR archive. First, the **"Coffee"** dataset consists of spectrographic data from food quality analysis. These series are relatively short (typically length 286) and exhibit specific transient shapes rather than long-term repeating cycles. Visual inspection reveals that this time series is non-stationary in the mean, containing distinct peaks corresponding to chemical features. Second, the **"PigCVP"** (Pig Central Venous Pressure) dataset represents a significantly longer physiological time series. It is characterized by a quasi-periodic structure driven by cardiac and respiratory cycles. Unlike the synthetic sine wave, the periodicity here is biological and subject to heart rate variability and measurement noise.

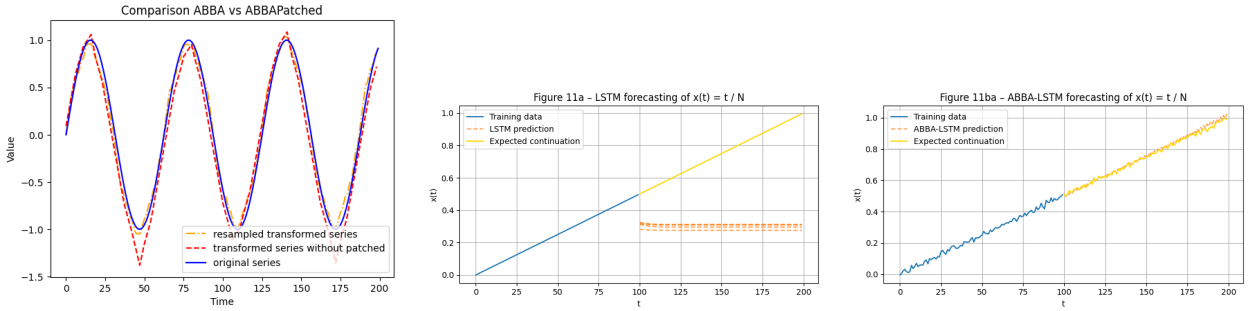
Before training, as neural networks are highly sensitive to the scale of input data, we applied Z-score normalization to all time series, defined as $x_{norm} = (x_t - \mu)/\sigma$. This ensures zero mean and unit variance, facilitating gradient descent convergence and making the ABBA tolerance parameter (tol) comparable across different datasets. Furthermore, our trend analysis indicated that raw LSTMs often fail on non-stationary data with linear trends, as they cannot forecast values outside the training range. We utilized the ABBA compression rate as a proxy for the structural complexity of the time series. For a fixed tolerance, simple signals like the sine wave yield short symbolic strings, whereas the PigCVP signal requires a larger alphabet size and longer strings to maintain reconstruction accuracy. To evaluate forecast quality, we rely on the Symmetric Mean Absolute Percentage Error (sMAPE) for numerical accuracy and Dynamic Time Warping (DTW) distance.

4 Results

In this section, we present the numerical results obtained on the three datasets described previously. We compare the performance of the proposed **ABBA-LSTM** approach against a standard baseline, the **raw LSTM**. The comparison focuses on two main aspects: forecasting accuracy (measured by RMSE and sMAPE) and computational efficiency (training time).

4.1 Validation on Synthetic Data

We first compared our three different implementations of the ABBA framework by reconstructing a sine wave, in order to validate that the patched version performs best both visually and in terms of DTW distance (see 1a and 3 in Appendix).We then reproduced the experiments from the article, highlighting two key behaviors. First, a raw LSTM struggles to predict values outside the range of the training data, whereas the ABBA–LSTM handles this more effectively, as illustrated on a synthetic near-linear time series in Figures 1b and 1c (designed to avoid degenerate single-symbol predictions). Second, the ABBA–LSTM better captures and predicts shapes, while the standard LSTM can struggle; this is demonstrated using the HouseTwenty dataset described earlier and shown in 4 in Appendix.



(a) Sine wave reconstruction (b) Raw LSTM forecast (c) ABBA–LSTM forecast

Figure 1: Comparison of reconstruction and forecasting behaviors

4.2 Performance on Real Datasets

We then studied the influence of selected parameters of the ABBA framework on sMAPE and DTW for the Coffee and PigCVP datasets presented in the article. The parameters considered include the compression tolerance, the minimum distance between two clusters α, the number of symbols used, and the lag. The quantitative results are summarized in Figures 5 and 6 in Appendix.

As shown in Figure 5, there is a clear trade-off with the compression tolerance. Low values ($tol < 0.01$) produce a very faithful representation with many symbols, making it hard for the LSTM to capture long dependencies and leading to overfitting, while high values ($tol > 0.1$) compress the signal too aggressively, losing important information and causing underfitting. We found an optimal balance around $tol = 0.01$, where the signal is simplified enough for the RNN to learn effectively, yet detailed enough to remain accurate. The behavior of α is similar: too large values generate too many symbols, too small values too few, so an intermediate value must be chosen.

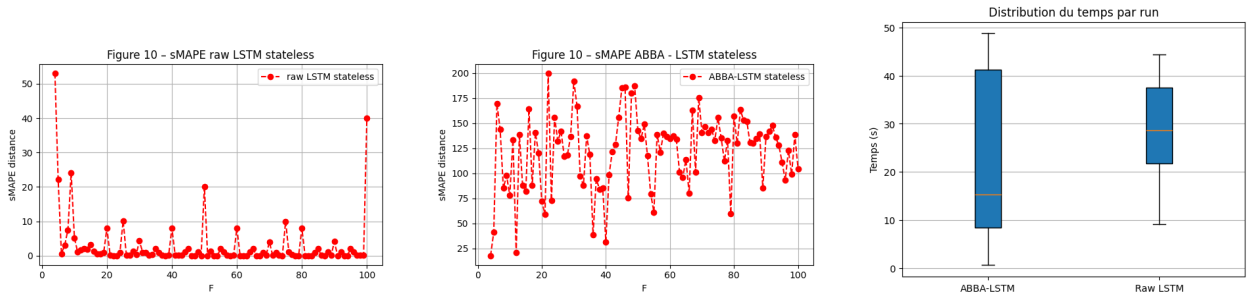
Overall, the results were not fully satisfactory. Experiments across different time series led us to conclude that selecting appropriate parameters is difficult, as performance is highly sensitive to these choices. Moreover, since the time series are short and become even shorter after compression, the training is particularly challenging. A frequent behavior we observed is the dominance of a single symbol, which leads the model to repeatedly predict this symbol, resulting in a trivial linear forecast. To mitigate these issues, we introduced dropout, adjusted the hidden layer dimensions and learning rate, and experimented with stochastic forecasting strategies instead of a softmax output, in order to introduce randomness into the predictions. Despite these efforts, per-

formance on the real datasets remained unsatisfactory. Consequently, we conducted additional experiments using simple sine waves with varying frequencies to compare the ABBA–LSTM approach with a raw LSTM model, focusing on training time consumption and sMAPE performance.

4.3 Comparison between raw-LSTM and ABBA-LSTM models

As previously stated, our primary objective is to evaluate whether the ABBA-LSTM model provides superior sMAPE scores and reduced runtimes compared to the baseline raw-LSTM model. To this end, we generated 100 sine waves with varying frequencies, each with a length of $N = 1000$ points. In this analysis, the *patience* parameter was set to 50, the *lag* to 50 for the raw-LSTM and 5 for the ABBA-LSTM (consistent with the article), and the learning rate to 0.0005.

We compared both approaches across the different frequencies using their sMAPE scores. The results indicate that the raw-LSTM model achieves better performance than the ABBA-LSTM approach, yielding a higher concentration of very low sMAPE scores. Consistent findings were observed when using the DTW distance. While these results do not imply that the baseline consistently outperforms the symbolic approach in every scenario, they suggest that the ABBA method may not be appropriate for trivial or highly simplistic datasets such as the one employed here.



(a) sMAPE Raw-LSTM (b) sMAPE ABBA-LSTM (c) Execution Runtimes

Figure 2: Comparison across 100 different frequencies.

However, when examining the runtime for each frequency in Figure 2c, the ABBA-LSTM method exhibits a significantly lower median runtime than the raw-LSTM, despite some occasional outliers. This observation strongly aligns with the conclusions of the original paper, confirming that, on average, ABBA-LSTM training and prediction times are substantially faster.

5 Conclusion

Through this project, we successfully implemented the full ABBA-LSTM pipeline, confirming the core claims made by the authors regarding computational efficiency. Our experiments on synthetic sine waves and real-world datasets from the UCR archive demonstrate that the symbolic representation provided by ABBA significantly reduces training and forecasting runtimes, often by a factor of two, by effectively compressing the temporal dimension of the data. However, our sensitivity analysis reveals that the model's performance is highly dependent on the choice of hyperparameters, such as the compression tolerance and clustering threshold, which can lead to issues ranging from over-compression to training instability on short symbolic strings. While the raw LSTM occasionally achieved higher numerical accuracy on simple periodic signals, the ABBA-LSTM proved to be a more robust forecaster for complex, non-stationary patterns.

References

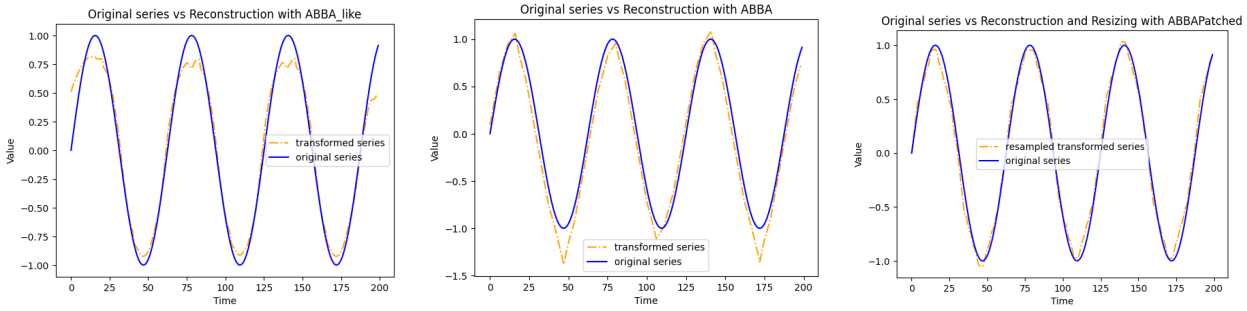
[1] Steven Elsworth and Stefan Güttel. Time series forecasting using lstm networks: A symbolic approach, 2020.

Appendices

LSTM architecture: At each time step t, the LSTM cell maintains a hidden state $h^{(t)}$ and a cell state $c^{(t)}$ updated via the standard gating mechanisms

$$f^{(t)} = \sigma(W_f \cdot [h^{(t-1)}, x^{(t)}] + b_f) \tag{1}$$
$$i^{(t)} = \sigma(W_i \cdot [h^{(t-1)}, x^{(t)}] + b_i) \tag{2}$$
$$\tilde{c}^{(t)} = \tanh(W_c \cdot [h^{(t-1)}, x^{(t)}] + b_c) \tag{3}$$
$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \tag{4}$$
$$o^{(t)} = \sigma(W_o \cdot [h^{(t-1)}, x^{(t)}] + b_o) \tag{5}$$
$$h^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \tag{6}$$

where σ is the sigmoid function, $\odot$ denotes element-wise multiplication, and $x^{(t)}$ is the input symbol representation.

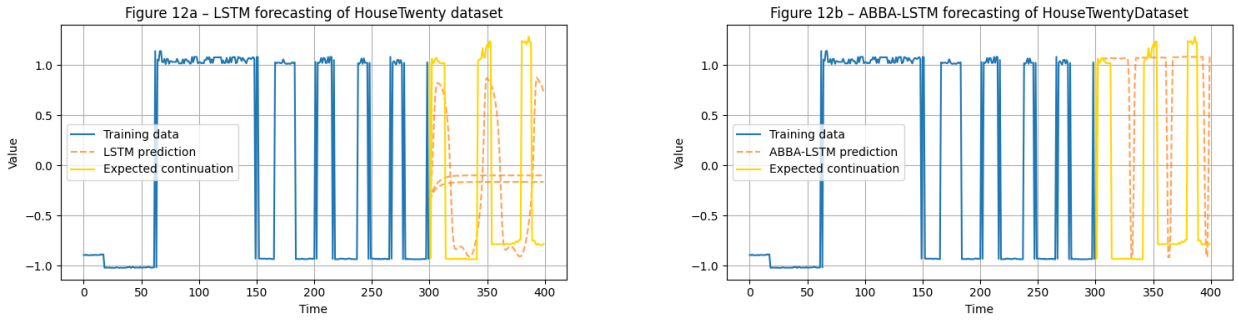


(a) Sine wave reconstruction with ABBA_like class, dtw= 15.5

(b) Sine wave reconstruction with ABBA class, dtw= 11.7

(c) Sine wave reconstruction with ABBAPatched class, dtw= 8.08

Figure 3: Sine waves reconstruction comparison with different ABBA methods



(a) HouseTwenty forecast with raw LSTM

(b) HouseTwenty forecast with ABBA-LSTM

Figure 4: Comparison of forecasting behaviors on HouseTwenty time series

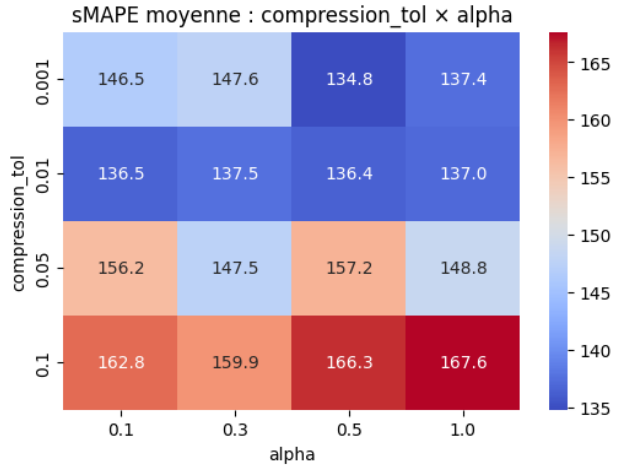
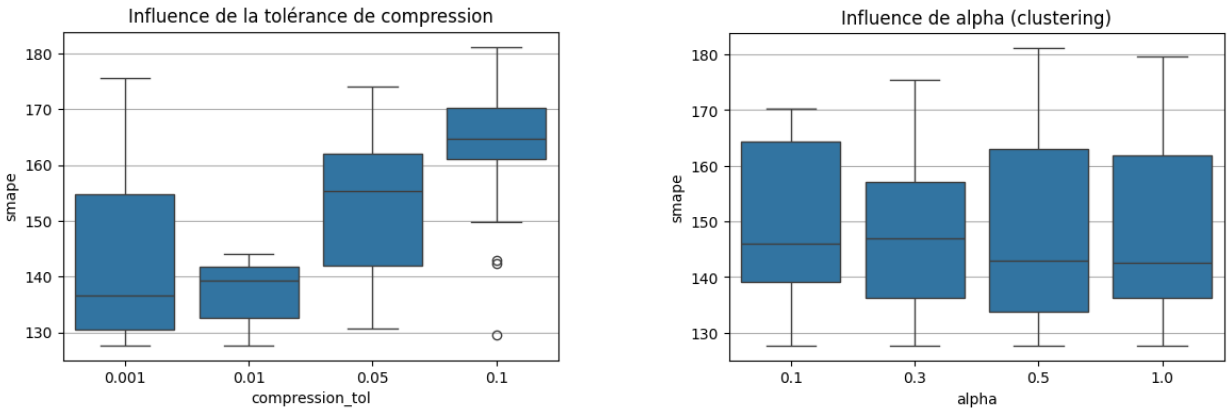


Figure 5: Sine wave reconstruction comparison



(a) Compression tolerance influence on sMAPE

(b) α influence on sMAPE

Figure 6: Compression tolerance and α influences on sMAPE