Seasonality Modeling through LSTM Network in Inflation-Indexed Swaps

Looking for a bridge between the traditional standard pricing approach and the new FinTech techniques

Pier Giuseppe Giribone
Department of Economics (DIEC)
University of Genoa
Genoa, Italy
e-mail: pier.giuseppe.giribone@economia.unige.it

Abstract—**An Inflation-Indexed Swap (IIS) is a derivative in which, at every payment date, the counterparties swap an inflation rate with a fixed rate. For the calculation of the Inflation Leg cash flows, it is necessary to build a mathematical model suitable for the Consumer Price Index (CPI) projection. For this purpose, quants typically start by using market quotes for the Zero-Coupon swaps in order to derive the future trend of the inflation index, together with a seasonality model for capturing the typical periodical effects. In this study, I propose a forecasting model for inflation seasonality based on a Long Short-Term Memory (LSTM) network: a deep learning methodology particularly useful for forecasting purposes. Thanks to its architecture, able to capture highly nonlinear relationships, and to the design of a careful training, able to satisfy both statistical and econometric features, the proposed methodology can be considered more accurate rather than the traditional one. As a result, the study shows how the CPI predictions, conducted using a FinTech paradigm, can be integrated in the respect of the traditional quantitative finance theory developed in this research field.**

Keywords-Inflation-Indexed Swap (IIS); Year-on-Year Inflation-Indexed Swap (YYIIS); Zero-Coupon Inflation-Indexed Swap (ZCIIS); Seasonality model; CPI bootstrap; Machine Learning (ML); Deep Learning; Long Short-Term Memory (LSTM) Network.

I. INTRODUCTION

Machine learning methodologies are increasingly spreading in the financial sector. Among the numerous examples of applications proposed by the literature, the most popular ones are mainly aimed at solving the following problems: input data quality [15], innovative algo-trading techniques [5], optimal portfolio management [7], pattern recognition and classification [11], financial time-series forecasting as an alternative to traditional econometric approaches, such as: Autoregressive Integrated Moving Average (ARIMA), Bayesian Vector AutoRegression (BVAR), Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) [13] [17].

It is more difficult to find evidence in literature of artificial intelligence methodologies applied to exotic financial instruments pricing or about the integration of traditional quantitative finance theory with the new FinTech methodologies. The traditional implementation regards the numerical solution of the so-called fundamental Black-Scholes-Merton PDE through Radial Basis Functions [4].

Only more recently, the application of Regressive Neural Networks together with the Monte Carlo method was suggested for evaluating early-exercise features in American and Bermuda option pricing in accordance with the Longstaff-Schwartz methodology [12]. This study aims to extend the existing literature concerning the integration of FinTech in Quantitative Finance through the design of a LSTM network for the seasonality modeling in inflation indexed swaps.

The paper is structured according to the following sections: the next part illustrates briefly the pricing framework; section 3 deals with the traditional standard method for the forecast of CPI values (trend + seasonality); section 4 describes the LSTM architecture; section 5 focuses on CPI projections (also called CPI bootstrap) and section 6 concludes with a real market case: the two methodologies are used for computing the fair-value for an Inflation-Indexed Swap and the model risk is quantified.

II. THE PRICING FRAMEWORK

An Inflation-Indexed Swap (IIS) is a swap deal in which, for each payment date, $T_1, \dots, T_M$, counterparty A pays to counterparty B the inflation rate in the considered period, while counterparty B pays to counterparty A the fixed rate.

The inflation rate is calculated as the percentage return of the Consumer Price Index (CPI) over the reference time interval. There are two main types of IIS traded on the market: the Zero-Coupon Inflation-Indexed Swap (ZCIIS) and the Year-on-Year Inflation-Indexed Swap (YYIIS) [2].

In a ZCIIS, at maturity date T_M, assuming $T_M = M$ years, counterparty B pays to counterparty A the fixed quantity:

$$N[(1+K)^M - 1] \quad (1)$$

where K and N are the fixed interest rate and the principal, respectively.

In return for this fixed payment, at the maturity date T_M, counterparty A pays to counterparty B the floating amount:

$$N\left[\frac{I(T_M)}{I_0} - 1\right] \quad (2)$$

In a YYIIS, for each payment date T_i, counterparty B pays to counterparty A the fixed amount:

$$N\varphi_i K \quad (3)$$

where φ_i is the year fraction of the fixed swap leg in the range $[T_{i-1}, T_i]$, $T_0 := 0$ and N is the principal of the deal.

Counterparty A pays to counterparty B the floating amount equals to:

$$N\varphi_i\left[\frac{I(T_i)}{I(T_{i-1})}-1\right] \quad (4)$$

ZCIIS and YYIIS are typically quoted in terms of the corresponding equivalent fixed rate K.

Based on these quotes and using stochastic calculus, pricing formulas can be derived for both classes of derivatives. Readers, interested in this quantitative financial part, can find the rigorous pricing formulas derivations in [2] [9] [10] and [14].

In particular Kazziha [10] derived the CPI forward values, $\mathfrak{I}_i$:

$$\mathfrak{I}_M(0)=\mathfrak{I}_{REF}(0)\cdot[1+K(T_M)]^M \quad (5)$$

where:

$\mathfrak{I}_{REF}(0)$ is the CPI reference value. It corresponds to the one set n months back in relation to the settlement date. Typically, the standard time lag is 3 months.

$K(T_M)$ is the Inflation Zero Swap Rate quoted on the market in correspondence to the maturity T_M.

III. CPI INDEX TRADITIONAL SIMULATION

Through (5), we are able to project the index values in the future according to the swap rates listed on the market following the pricing framework. Since the frequency with which the index is published is monthly, it is necessary to provide a simulation of the CPI with such periodicity [3].

The missing curve points are therefore estimated by adding the logarithm of the monthly increase between a calculated value $\mathfrak{I}_M(0)$ and its subsequent value $\mathfrak{I}_{M+1}(0)$:

$$\Delta\mathfrak{I}_M=\frac{\ln\left(\frac{\mathfrak{I}_{M+1}(0)}{\mathfrak{I}_M(0)}\right)}{12\cdot\tau} \quad (6)$$

where τ is the time interval expressed in year fraction between $\mathfrak{I}_M(0)$ and $\mathfrak{I}_{M+1}(0)$.

The points making up the simulated curve of the consumer price index are defined by the formula:

$$\mathfrak{I}_{i+1}=\mathfrak{I}_i\exp(\Delta\mathfrak{I}_M+\mathfrak{R}_M),\ \mathfrak{I}_M(0)\le\mathfrak{I}_i\le\mathfrak{I}_{M+1}(0) \quad (7)$$

The standard methodology, suggested by the main benchmark info provider pricing modules, takes into account the index seasonality algebraically adding the normalized residuals $\mathfrak{R}_M$ obtained from the historical values of the CPI, in accordance with the expression (8):

$$\mathfrak{R}_M=\frac{\sum_{i=1}^{seasyear}\ln\left[\frac{\mathfrak{I}_{i+1}^{Monthly}}{\mathfrak{I}_i^{Monthly}}\right]}{seasyear}-\frac{\sum_{i=1}^{12\cdot seasyear}\ln\left[\frac{\mathfrak{I}_{i+1}^{Monthly}}{\mathfrak{I}_i^{Monthly}}\right]}{12\cdot seasyear} \quad (8)$$

where $\mathfrak{R}_M$ are the standardized residuals obtained from the effect of seasonality over $seasyear$ years. The first contribution is the logarithmic variation of the CPI values on the considered month; the second one represents the overall logarithmic variation recorded in the time period considered for seasonality. The objective of this study is to propose a deep learning methodology (LSTM network) able to simulate the seasonality of the inflation index. In this way, in addition to introducing a more robust and flexible econometric methodology than the standard one, the integration between the classic quantitative finance theory together with the Fintech paradigms can be considered an interesting feature [1]. In fact, the determination of the swap fair value is implemented by applying the formulas described above for the ZCIIS and YYIIS and therefore in total agreement with canonical principles; moreover, a Long Short-Term Memory network will be implemented for a more reliable simulation of the CPI seasonality. The next section deals with the explanation of the architecture and the training phase for the implemented LSTM network.

IV. LSTM NETWORK ARCHITECTURE AND TRAINING

LSTM networks are also able to learn long-term relationships between the time intervals of a time series, therefore without the need to pre-set the number of time lags, as occurs in other dynamic recurrent networks, such as Nonlinear AutoRegressive (NAR) and Nonlinear Auto-Regressive with exogenous variables (NARX) [6].

A common LSTM unit is composed of a cell, an input gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. Intuitively, the cell is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit [8]. The activation function of the LSTM gates is often the logistic sigmoid. Figure 1 shows how the flux of a data sequence Y with C features (or channels) of length S has been processed into a LSTM layer. In the block diagram, h_t and $c(t)$ are, respectively, the output (also known as hidden state) and the cell state at time t.

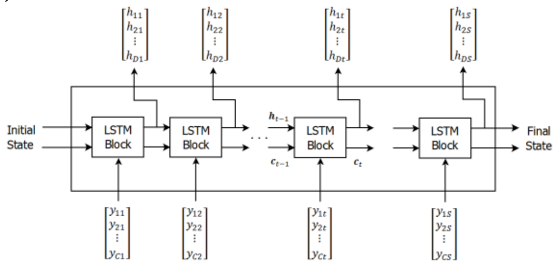
$[h_{11}, h_{21}, \vdots, h_{D1}]$
$[h_{12}, h_{22}, \vdots, h_{D2}]$
$[h_{1t}, h_{2t}, \vdots, h_{Dt}]$
$[h_{1S}, h_{2S}, \vdots, h_{DS}]$
Initial State
LSTM Block
LSTM Block
h_{t-1}
c_{t-1}
LSTM Block
c_t
LSTM Block
Final State
$[y_{11}, y_{21}, \vdots, y_{C1}]$
$[y_{12}, y_{22}, \vdots, y_{C2}]$
$[y_{1t}, y_{2t}, \vdots, y_{Ct}]$
$[y_{1S}, y_{2S}, \vdots, y_{CS}]$

Figure 1. LSTM network architecture.

The first LSTM block uses the initial state of the network and the first time-step of the sequence in order to compute the first output and the first update of the cell state. At time t, the block uses the current state of the network (c_{t-1}, h_{t-1}) and the next step of the sequence for estimating the output and updating the current state of the cell c_t. The layer state is characterized by the hidden state (also known as the output state) and the cell state. The hidden state at time step t contains the output of the LSTM layer for the current time step. The cell state contains the information learnt in the previous steps. For each time step, the layer adds or removes information from the cell state. The layer controls these updates using gates. The following components control the cell state and the hidden state of the layer [8]:

- Input gate (i): Control level of cell state update.

- Forget gate (f): Control level of cell state reset (forget).
- Cell candidate (g): Add information to cell state.
- Output gate (o): Control level of cell state added to hidden state.

Figure 2 shows how the gates (i, f, g, o) process the signal at time t.

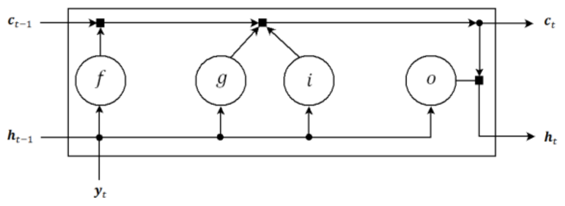

Figure 2. Signal processed by the gates (i, f, g, o).

In a LSTM, the parameters that are subjected to calibration are: the input weights (W), the recurrent weights (R) and the biases (b) [6]. W, R and b are the arrays built through the concatenations of such parameters for each component: $W = (W_i, W_f, W_g, W_o)^{\top}$, $R = (R_i, R_f, R_g, R_o)^{\top}$ and $b = (b_i, b_f, b_g, b_o)^{\top}$ where i , f , g and o denote the input gate, the forget gate, the cell candidate and the output gate, respectively.

At time step t, the cell state is given by:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (9)$$

where $\odot$ is the Hadamard product operator.

At time step t, the hidden state is given by:

$$h_t = o_t \odot \sigma_c(c_t) \quad (10)$$

where σ_c is the activation function of the state (typically a hyperbolic tangent).

The following equations define the components at time step t:

- Input gate (i):

$$i_t = \sigma_g(W_i y_t + R_i h_{t-1} + b_i) \quad (11)$$

- Forget gate (f):

$$f_t = \sigma_g(W_f y_t + R_f h_{t-1} + b_f) \quad (12)$$

- Cell candidate (g):

$$g_t = \sigma_c(W_g y_t + R_g h_{t-1} + b_g) \quad (13)$$

- Output gate (o):

$$o_t = \sigma_g(W_o y_t + R_o h_{t-1} + b_o) \quad (14)$$

σ_g is the activation function of the gate, typically a sigmoid.

LSTMs are supervised networks, as a result, after the design of the model, it is essential to implement a robust algorithm for the training phase. This is the part in which the designer decides how many neurons must be implemented in order to make reliable predictions. In order to obtain valid models for forecasting purposes it is necessary to conduct statistical and econometric tests. The objective of the first kind of test is to tune the LSTM in order to have a good fitting of the training dataset. The gap between the target and the model output is reduced through an ADAM optimizer as the network training process progresses, so it may happen that the estimated relationship returns a perfect fit of the sampled data (in-sample), making vain the attempt at generalization, fundamental for making the network capable of processing different data (out-of-sample). For this reason and especially in the field of deep learning where there is a huge number of parameters to tune in order to capture highly non-linear relationships, special measures for avoiding overfitting must be taken into consideration. As a result, the first intervention, shared also with traditional recurrent networks, such as NAR and NARX, is to work directly on the dataset through a random-splitting method.

The data set configuration used for the network is:

- 70% of the set will form the training set, thus the optimization will be carried out with respect to its loss function (J) only.
- 15% of the set will be assigned to the validation set, thus, despite the weights are updated with respect to the train set, the algorithm saves the weights that minimize J on the validation set, in order to avoid data overfitting and trying to reach a good generalization.
- 15% of the data set will form the test set, so that the network performance can be measured on data that it has never seen before, as the ultimate objective of a neural network user is to employ the network on completely new data.

The second kind of statistical measures, which are traditionally applied in the field of deep learning, work directly on the network. The implemented measures can be summarized as follows:

- Adding a term to the traditional loss function (RMSE) which put in a penalty (the λ coefficient) if a further weight (ω) associated to an arch has been activated: $J = RMSE + \frac{1}{2}\lambda\|\omega\|^2$
- Dropout, which is a technique consisting of training only a group of randomly selected neurons rather than the entire network: a percentage (a popular choice is 25%) determines how many neurons to choose and the remaining ones are deactivated. Since the neurons and the relative weights are continuously modified, it is thus possible to avoid overfitting.

These precautions are thus implemented in the forecaster in order to have a reliable fitting.

Given that the objective is to perform a prediction of the most reasonable CPI projections, the second test has an econometric nature. It is based on the verification of the autocorrelation error absence so that the model error is unstructured and the predicted values can be econometrically reliable.

V. COMPARISON BETWEEN STANDARD AND LSTM TECHNIQUES FOR THE CPI PROJECTION

In order to compare the standard inflation bootstrap methodology with the LSTM approach, we use the market data retrieved from Bloomberg on 30th June 2020. Swap rate values, $K(T_M)$, quoted by the market at the reference date are reported in Table I, together with the estimation of the CPI projections, $\Im_M(0)$ and the $\Delta\Im_M$, according to (5) and (6). The estimation of $\Delta\Im_M$ is useful in order to have the inflation values expressed on a monthly basis [3].

TABLE I. $K(T_M), \Im_M(0)$ AND $\Delta\Im_M$ (30TH JUNE 2020)

T_M	***Mid Price*** $K(T_M)$	$\Im_M(0)$	$\Delta\Im_M$
1	-0.071	104.69561	0.0382
2	0.19375	105.17638	0.0543
3	0.347	105.86444	0.0787
4	0.497	106.86841	0.0721
5	0.57125	107.79688	0.0804

According to (7), this information allow us to project the CPI values for the next years using a market-oriented approach without taking into account the seasonality. In order to add this essential contribution for the forecast into the model, we have to consider the monthly normalized residuals, $\Re_M$, calculated starting from the past CPI realizations. The traditional way to implement this task is to apply (8).

Using the traditional market standard preference to consider the previous five years of the CPI time-series, we get the following $\Re_M$, from January to December: -0.011959, 0.001761, 0.008373, 0.00266, 0.000828, 0.00018,-0.005856, 0.000625, 0.002751, 0.00105, -0.002176, 0.001764.

Applying recursively (7), the projections for the CPI are obtained for the following years. These simulations are reported in Figure 3 together with the past values.

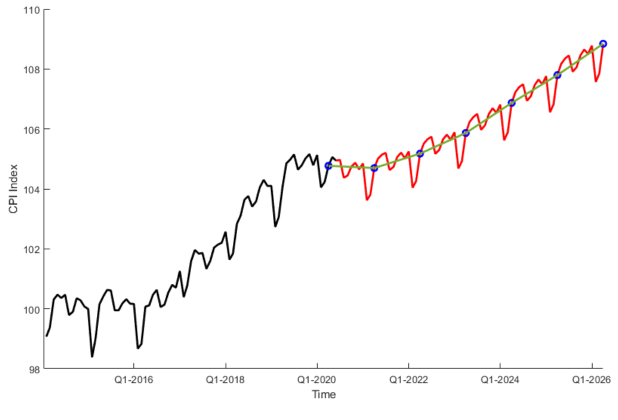

Figure 3. CPI time-series and its projection (traditional methodology). Black line: past CPI values. Green line: CPI projection without seasonality. Red Line: CPI projection with seasonality (traditional model). Blue points: market-implied CPI estimation

- the black line represents the past five years CPI values used for the estimation of the seasonality effect.
- the green line represents the CPI projections without seasonality: it connects the blue dots which are the $\Im_M(0)$ whose estimations are strictly connected to the $K(T_M)$ quotation.
- the red line represents the projections of the CPI index taking into account the seasonality through the monthly annualized residuals $\Re_M$.

The idea is to use a LSTM network with the aim of providing a better model for the seasonality. For the training set, we use the monthly return of the index computed in the last 5 years: $\ln\left[\frac{\Im_{i+1}^{Monthly}}{\Im_{i}^{Monthly}}\right]$, according to the market standard convention. The number of hidden units in the LSTM block is tuned in function of the performances recorded by the network. Using a layer made by 100 neurons, adopting an ADAM optimizer and implementing all the described techniques in order to avoid overfitting, we can achieve excellent results in the training phase [6]. From a statistical point of view, we obtain an R^2 close to 1, as a result the fitting over the historical time series is extremely good. From an econometric point of view, the auto-correlation error for the tuned model has been kept under an acceptable threshold for the non-zero lags [16], with a confidence interval equal to 95%.

Having checked the forecasting reliability of the LSTM network, we proceed to compute the following 6 years returns (72 values). Figure 4 and Figure 5 show the difference between the two approaches: the black line represents the realized past returns of the last five years and the red line represents the forecasted returns.

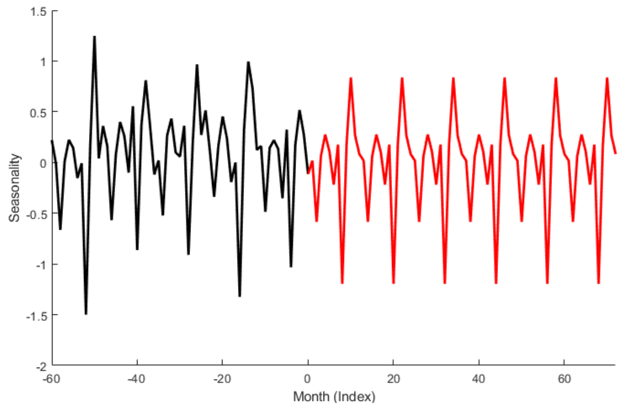

Figure 4. Historical and prospective seasonality estimation using the standard technique. Black line: realized past returns. Red line: forecasted returns

It is sufficient to look at the figures to realize that the red line (i.e. the projected time-series) obtained from the traditional method has a behavior which is too simplified. In fact, it is based on the estimation of the twelve normalized residuals of the previous months which are repeated equal for the future values. Implementing a properly trained LSTM allows to use a model able to capture highly nonlinear relationship among the time-series in accordance with the rigorous statistical and econometric tests [16] described in Section IV.

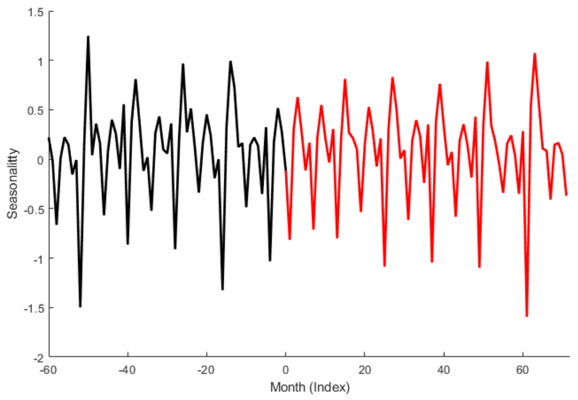

Figure 5. Historical and prospective estimation for seasonality using Deep Learning. Black line: realized past returns. Red line: forecasted returns

As a result, facing the forecasting problem with the FinTech approach, the red line has a more realistic forward-looking behavior thanks to both the advanced technology (deep learning) and the careful tuning. As we will see in the market case, which regards the pricing of a YYIIS, these differences in the simulation of the seasonality cause an impact on the derivative fair-value that is not always negligible.

VI. Market Case: YYIIS pricing

In this section, we proceed with the valorization of a YYIIS using the two approaches previously described. The main financial characteristics are reported in Table II. The valuation date of the "In Arrears" swap is 30th June 2020, as a result we use the historical and prospective inflation data already computed in the previous sections. Regarding the discount curve we use, according to the new benchmark standard for collateralized derivatives, the EUR OIS ESTR term structure. As a result, zero rates and discount factors used for pricing are those implied from the new market benchmark curve. Using the pricing framework described in section II, we proceed with the estimation of the future cash-flows for the swap and then we go through the discounting process for obtaining the NPVs for the two legs. The difference between the two NPVs gives the price of the swap.

TABLE II. YYIIS financial characteristics

	Receiving Leg	*Paying Leg*
Leg Type	Y-o-Y Inflation	Fixed
Notional	10 MM	10 MM
Currency	Euro	Euro
Index	CPTFEMU Index	Fixed Coupon: 0.5%
Effective Date	30th June 2020	30th June 2020
Maturity Date	30th June 2026	30th June 2026
Lag	3 Month	-
Interpolation	Monthly	-
Spread	0	-
Reset Frequency	Semi-Annual	-
Payment Freq.	Semi-Annual	Annual
Day Count	ACT/ACT	ACT/ACT
Discount Curve	EUR-OIS-ESTR	EUR-OIS-ESTR

The discounted Cash Flows for the fixed paying leg of the swap are equal to -306,314.62 Euro (Table III).

TABLE III. YYIIS Paying Leg

Payment Date	*Payment*	*Discount Rate*	*Present Value*
06/30/2021	-49,930.76	1.006019	-50,231.30
06/30/2022	-50,000.00	1.012638	-50,631.89
06/30/2023	-50,000.00	1.019137	-50,956.87
06/30/2024	-49,796.02	1.025040	-51,042.90
06/30/2025	-50,203.98	1.030223	-51,721.29
06/30/2026	-50,000.00	1.034607	-51,730.37

The discounted Cash Flows for the inflation-indexed receiving leg of the swap using the standard seasonality approach are equal to +391,740.5 Euro (Table IV).

TABLE IV. YYIIS Receiving Leg (standard approach)

Date	*Reset CPI*	*Payment*	*Discount*	*PV*
12/31/2020	104.74729	-2,185.66	1.002913	-2192.02
06/30/2021	104.69561	-4,894.44	1.006019	-4923.89
12/31/2021	105.06038	35,066.31	1.009331	35393.52
06/30/2022	105.17638	10,944.48	1.012638	11082.8
12/30/2022	105.6452	44,597.45	1.015884	45305.83
06/30/2023	105.86444	20,674.18	1.019137	21069.82
12/29/2023	106.49159	58,904.25	1.022122	60207.33
06/28/2024	106.86841	35,225.72	1.02504	36107.77
12/31/2024	107.45914	56,181.35	1.027729	57739.21
06/30/2025	107.79688	31,122.41	1.030223	32063.02
12/31/2025	108.44679	60,603.27	1.032507	62573.3
06/30/2026	108.84187	360,65.68	1.034607	37313.81

The discounted Cash Flows for the inflation-indexed receiving leg of the swap using the deep learning architecture are equal to +371,023.4 Euro (Table V).

It is interesting to highlight that, in correspondence with the dates where the CPI values can be directly implied by the market using (5), both methodologies are consistent with the values reported in Table I. This shows a good integration between traditional quantitative finance principles and new FinTech paradigms.

TABLE V. YYIIS RECEIVING LEG (LSTM APPROACH)

Date	*Reset CPI*	*Payment*	*Discount*	*PV*
12/31/2020	104.80403	3274,34	1.002913	3283.87
06/30/2021	104.69561	-10265.09	1.006019	-10326.88
12/31/2021	104.75465	5683.47	1.009331	5736.51
06/30/2022	105,17638	39847.66	1.012638	40351.25
12/30/2022	105.33777	15375.28	1.015884	15619.50
06/30/2023	105.86444	49737.38	1.019137	50689.20
12/29/2023	106.18170	29841.55	1.022122	30501.70
06/28/2024	106.86841	64287.66	1.02504	65897.43
12/31/2024	107.14643	26480.24	1.027729	27214.51
06/30/2025	107.79688	60025.26	1.030223	61839.40
12/31/2025	108.13120	31220.98	1.032507	32235.88
06/30/2026	108.63801	46376.07	1.034607	47981.01

According to the traditional pricing approach, the fair value for the analyzed YYIIS is +85,425.87. On the other hand, if we would have used the proposed and more advanced approach, its value would have been +64,708.77.

The gap between the values from the two pricing methodologies is equal to 20,717.1, an amount that can be considered significant enough for causing a percentage error higher than 20% compared to the Mark to Market of the analyzed derivative.

VII. CONCLUSION

This study shows how a Deep Learning methodology can be usefully implemented in a pricing framework aimed at determining the fair value of derivatives linked to the inflation index. The Long Short-Term Memory has allowed to identify the effect of seasonality more reliably than the traditional standard methodology. In fact, the proposed technique is able to simulate the future values of the time series by applying the described rigorous statistical and econometric tests, reasonably guaranteeing the reliability of the forecast. On the contrary, the traditional approach, based on the estimation of the historical normalized residuals, does not consider these important tests and it is not able to capture highly nonlinear relationships as a LSTM network does. It is particularly interesting considering how artificial intelligence paradigms can be integrated with traditional pricing methodologies in the quantitative finance field. For the continuation of the study, it is interesting to apply the suggested technology to derivatives written on an underlying which differs from inflation, where the seasonality modeling is of fundamental importance, such as commodity and energy derivatives.

ACKNOWLEDGMENT

I would like to thank Banca CARIGE for providing me market data through the Bloomberg® platform and the computational power needed for my researches over these past twelve years.

REFERENCES

[1] S. Bonini, G. Caivano, P. Cerchiello and P. G. Giribone, “Artificial Intelligence: Applications of Machine Learning and Predictive Analytics in Risk Management” AIFIRM (Italian Association of Financial Industry Risk Managers) position paper, N. 14, pp. 1–164, 2019.

[2] D. Brigo and F. Mercurio, “Interest Rate Models: Theory and Practice with smile, inflation and credit” Springer Finance, 2006.

[3] O. Caligaris and P. G. Giribone, “Modeling seasonality in inflation indexed swap through machine learning techniques: analysis and comparison between traditional methods and neural networks” Risk Management Magazine, vol. 13, N. 3, pp. 37–53, December 2018.

[4] R. Company, V. N. Egorova, L. Jodar and F. Soleymani, “A local radial basis function method for high-dimensional American option pricing problems” Mathematical Modelling and Analysis, vol. 23, N. 1, pp. 117–138, 2018.

[5] M. L. De Prado, “Advances in Financial Machine Learning”, Wiley, 2018.

[6] M. de Simon-Martin et al., “Electricity Spot Prices Forecasting for MIBEL by using Deep Learning: a comparison between NAR, NARX and LSTM networks” 20th International Conference on Environment and Electrical Engineering (IEEE-EEEIC) 2020 Proceedings, 11th June 2020

[7] J. B. Heaton, N. Polson and J. H. Witte, “Deep Learning for Finance: Deep Portfolios” Applied Stochastic Models in Business and Industry, vol. 33, N. 1, pp. 3-12, 2017.

[8] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory” Neural Computing, vol. 9, N. 8, pp. 1735–1780, 1997.

[9] R. Jarrow and Y. Yildirim, “Pricing Treasury Inflation protected securities and related derivatives using an HJM model” Journal of Financial and Quantitative Analysis, vol. 38, N. 2, pp. 337–358, June 2003.

[10] S. Kazziha, “Interest rate models, inflation-based derivatives, trigger notes and cross-currency swaptions”, PhD Thesis, Imperial Collage of Science, Technology and Medicine, London, 1999.

[11] P. Kim, “MATLAB Deep Learning with Machine Learning, Neural Networks and Artificial Intelligence” Apress, 2017.

[12] J. Lelong and B. Lapeyre, “Longstaff Schwartz algorithm and Neural Network regression” Advances in Financial Mathematics, Paris, 2020, unpublished.

[13] S. Mammadi, “Financial time series prediction using artificial neural network based on Levenberg-Marquardt algorithm” Procedia Computer Science, vol. 120, pp. 602–607, 2017.

[14] F. Mercurio, “Pricing inflation-indexed derivatives” Quantitative Finance, vol. 5, N. 3, pp. 289–302, June 2005.

[15] V. Pendyala, “Veracity of Big Data: Machine Learning and other approaches to verifying truthfulness” Apress, 2018.

[16] R. S. Tsay, “Analysis of Financial Time series” Third Edition, Wiley, 2010.

[17] C. Yanui, H. Kaijian and K. F. Geoffrey, “Forecasting crude oil prices: a deep learning based model” Procedia Computer Science, vol. 122, pp. 300–307, 2017.