

# Forex Forecasting

Combining statistical methods with neural networks

---

Eustathios Kotsis <sup>1</sup>   Darmanis Michael <sup>1</sup>   Vasilios Venieris <sup>1</sup>

September 27, 2023

<sup>1</sup>National and Kapodistrian University of Athens

# Smoothed Convolutional Neural Network (S-CNN)

---

# S-CNN: Origin and basic idea

- Implementation based on the paper “Time-series analysis with smoothed Convolutional Neural Network”[3] (no available code) .
- Model is univariate, multi-step.
- Simple exponential smoothing (remove outliers, average) + CNN.

# S-CNN: Simple Exponential Smoothing i

$$s_t = \alpha X_t + (1 - \alpha) s_{t-1} = s_{t-1} + \alpha (X_t - s_{t-1})$$

- For  $\alpha$  values close to 1 we get very little influence of the previous smoothed value.
- For  $\alpha$  close to 1 we also see that the previous unsmoothed series observation influences the result more than the smoothed values.
- For the calculation of  $\alpha$ , since there is no golden rule, we adopted the  $\alpha$  value from the aforementioned publication.

## S-CNN: Simple Exponential Smoothing ii

- In particular the authors argue that  $\alpha$  should be dependent on the dataset and must be of course between 0 and 1. Also the average value of the series should be less than the difference between max and min.

So we end up with the following formula

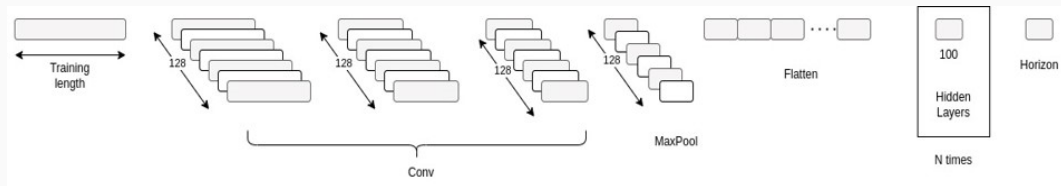
$$\alpha_{\text{opt}} = \frac{(X_{\max} - X_{\min}) - \frac{1}{n} \sum_{t=1}^n X_t}{X_{\max} - X_{\min}},$$

and the simple smoothing becomes

$$S_t = S_{t-1} + \frac{(X_{\max} - X_{\min}) - \frac{1}{n} \sum_{t=1}^n X_t}{X_{\max} - X_{\min}} (X_t - S_{t-1}).$$

# S-CNN: Architecture

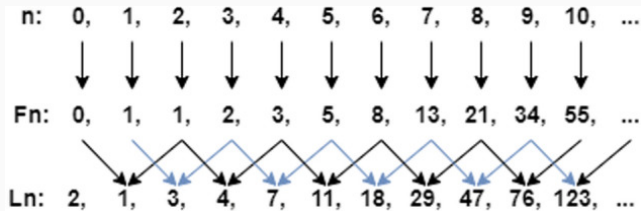
Basic architecture from Wibawa et al.[3] and experimented with different number of hidden layers.



Kernel size: 1x2, Stride: 1x1

## HOW TO PICK THE NUMBER OF HIDDEN LAYERS?

We used the Lucas numbers suggested. They derive from the Fibonacci Sequence if instead of adding every 2 successive observations we skip the intermediate and add the K with the K+2 together.



## S-CNN: Training ii

Model yielded the lower MSE, for the most stable currencies, for 76 layers of training.

**Table 1:** Training time for different numbers of hidden layers

Number of Hidden Layers	Training Time (hh:mm:ss)
3	0:27:43.15
11	0:34:27.77
47	1:05:21.78
76	1:28:47.70



### TRAINING STEPS...

- Split the dataset into series of different frequencies.
- For each time series we normalise + smooth.
- Feed the output into the convolutional model and train the model to predict a horizon of  $K$  steps for this frequency, where  $K$  is a model's designers choice.

## TRAINING HISTORY.

In order to balance training with only recent data, or feeding with irrelevant outdated history, we train with a list of possible training lengths.

- Daily: [14, 20, 240]
- Weekly: [52, 13, 26]
- Monthly: [24, 18]
- Quarterly: [12, 8]
- Yearly: [6]

## TRAIN/VALIDATION/TEST SPLIT.

- We focused on data after 2010 since in 2008 – 2010 there was the European debt crisis. A plot from the data can show that the economy and the currency fell rapidly in most countries in the European continent and in many other countries.
- For the frequencies except Yearly, we trained with splits of 80%-20%, 70%-30%, and approximately 60%-40% for quarterly (training/validation).

- We split the dataset accordingly and fed the created datasets to the generators.
- The models were trained for 200 epochs, using the MSE loss function and the Adam optimiser.

## S-CNN: Results i

For stable currencies, longer training tends to yield more accurate predictions.

**Table 2:** sMAPE for Daily with 76 Hidden Layers (Biggest Lucas Number Tested)

	SMAPE	
	Training length 20 days	Training length 240 days
CAD	104.24	26.63
AUD	226.39	79.63
DKK	1.357	0.937

## S-CNN: Results ii

Unpredictable currencies benefit from short-term history and possibly higher  $\alpha$  for forecasting. More hidden layers may overfit. Fewer hidden layers excel for daily predictions of unstable currencies.

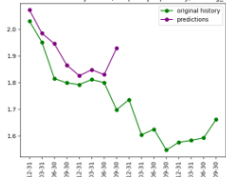
**Table 3:** sMAPE for 76 Hidden Layers

	sMAPE			
	T.L. 20 days	T.L. 240 days	T.L. 8 quarters	T.L. 12 quarters
USD	59.67	138.76	72.69	82.30

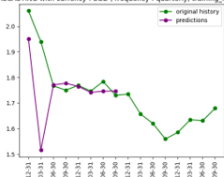
# S-CNN: Results iii

Quarterly and monthly predictions outperform daily ones for stable currencies. Daily and weekly forecasts show many fluctuations and are less accurate.

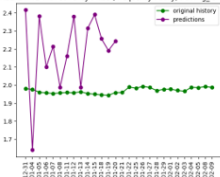
FORECASTING with currency : NZD , frequency : quarterly, training\_length :



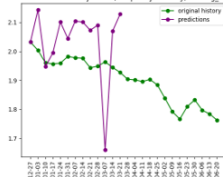
FORECASTING with currency : SGD , frequency : quarterly, training\_length :



FORECASTING with currency : NZD , frequency : daily, training\_length : 20



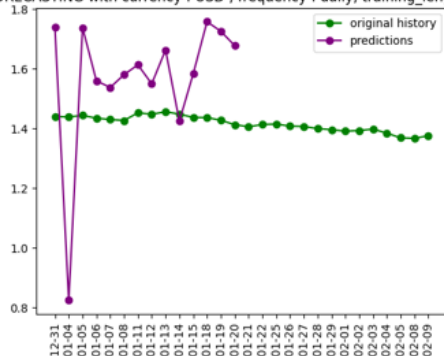
FORECASTING with currency : NZD , frequency : weekly, training\_length : 5:



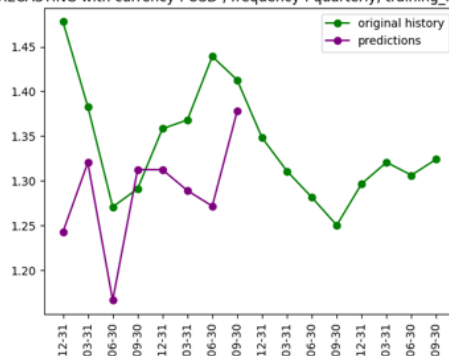
# S-CNN: Results iv

Model struggles with unpredictable currencies like *USD* and *GBP*.

FORECASTING with currency : USD , frequency : daily, training\_length : 14



FORECASTING with currency : USD , frequency : quarterly, training\_length : 1





# Exponentially Smoothed Recurrent Neural Network

---

## ES-RNN: Origin and basic idea

- Implementation based on Slawek Smyl[2], 2018  
*<https://eng.uber.com/m4-forecasting-competition/>*,  
Winner of M4 competition[1] (code available in C++)
- Use the Hotl's-Winters equations[4], and replace the trend with an RNN.

# ES-RNN: Holts-Winters model

Break down the problem of forecasting into three distinct equations:

- level,  $l_t(\alpha)$
- trend,  $b_t(\beta)$
- seasonality,  $s_t(\gamma)$

Optimisation problem with respect to  $\alpha, \beta, \gamma$ .

**ISSUE: FORMULATION WORKS FOR A LINEAR TREND.**

So plug in RNN for the trend component  $b_t(\beta)$ .

## ES-RNN: Holts-Winters model (multiplicative)

$$l_t = \alpha\left(\frac{y_t}{s_{t-m}}\right) + (1 - \alpha)l_{t-1}b_{t-1} \quad (1)$$

$$b_t = \beta\left(\frac{l_t}{l_{t-1}}\right) + (1 - \beta)b_{t-1} \quad (2)$$

$$s_t = \gamma\frac{y_t}{l_{t-1}b_{t-1}} + (1 - \gamma)s_{t-m} \quad (3)$$

$$\hat{y}_{t+h} = l_t b_t^h s_{t-m+h_m^+} \quad (4)$$

Equation 2 is replaced by an RNN from the model as follows:

$$\hat{y}_{t+1\dots t+h} = RNN(X_t) * l_t * s_{t+1\dots t+h} \quad (5)$$

$$x_i = \frac{y_i}{l_t s_i} \quad (6)$$

## ES-RNN: Holts-Winters model (additive)

$$l_t = \alpha(y_t - s_{t-m}) + (1 - \alpha)(l_{t-1} + b_{t-1}) \quad (7)$$

$$b_t = \beta(l_t - l_{t-1}) + (1 - \beta)b_{t-1} \quad (8)$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1 - \gamma)s_{t-m} \quad (9)$$

$$\hat{y}_{t+h} = l_t + h \cdot b_t + s_{t-m+h(\bmod m)} \quad (10)$$

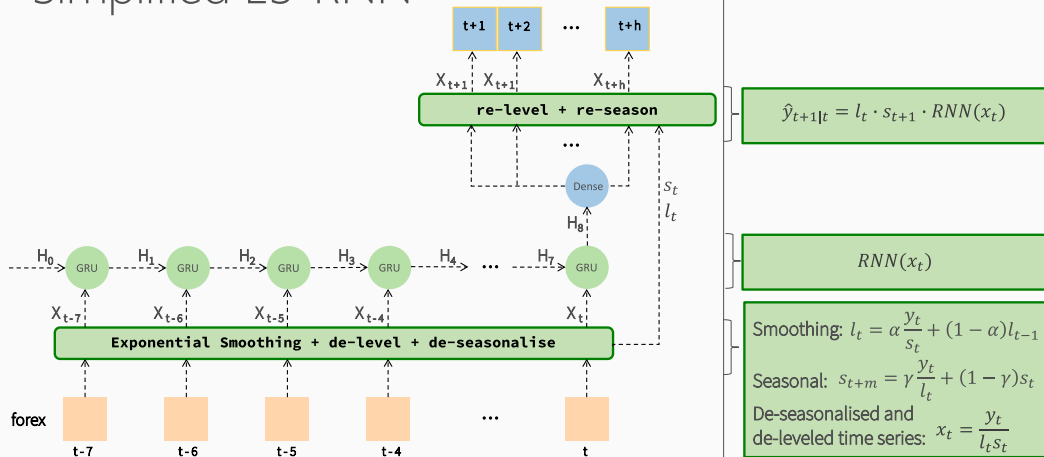
Equation 8 is replaced by an RNN in the model as follows:

$$\hat{y}_{t+1\dots t+h} = RNN(X_t) + l_t + s_{t+1\dots t+h} \quad (11)$$

$$x_i = y_i - l_t - s_i \quad (12)$$

# ES-RNN: Architecture

## Simplified ES-RNN



# ES-RNN: Training

Used a sliding window. Input and output windows were of constant size. Output size matched the prediction horizon, while input size determined heuristically.

## LOSS FUNCTION?

Considered the L1 difference as similar enough to sMAPE; both are mean absolute differences between forecasted and actual values.

## EXPERIMENTED WITH

- Both multiplicative and additive Holts-Winters formulations.
- A simple GRU with various hidden state features (8, 16, 32), i.e. input-window size.

Multiplicative Holts-Winters is better suited; best results were achieved using an input window of 8.

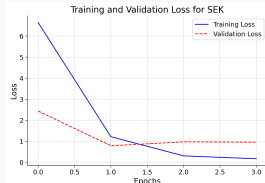
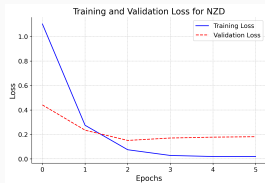
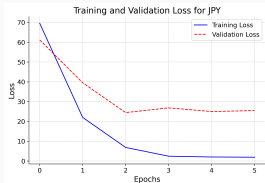
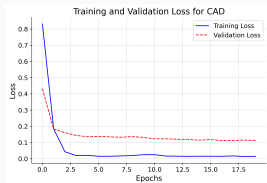
**Table 4:** Averaged sMAPE(%) values for daily and Weekly Frequencies

	Daily	Weekly
<i>Additive Holts-Winters</i>		
Input=8	23.45	17.89
Input=16	36.72	21.63
Input=32	28.83	19.42
<i>Multiplicative Holts-Winters</i>		
Input=8	13.72	7.96
Input=16	18.25	11.37
Input=32	16.92	9.84



# ES-RNN: Results ii

L1 as the training loss function created a positive bias.



# Comparison with Benchmark

---

# Average performance estimators

**Table 5:** Averaged performance estimators for daily and weekly frequencies

	Daily		Weekly	
	<i>sMAPE</i> (%)	<i>MSE</i>	<i>sMAPE</i> (%)	<i>MSE</i>
S-CNN	59.70	0.006	81.45	0.010
ES-RNN	13.72	560.580	7.96	323.021
V-AR	0.83	0.001	2.24	0.000

# Conclusions



---



## Conclusion and Key Findings i

- Explored the synergy between neural networks and traditional statistical models for forex forecasting.
- ES-RNN and S-CNN were both outperformed by Vector Autoregression (V-AR).
- Identified a positive bias issue in the ES-RNN due to an ineffective training loss function.
- S-CNN struggled with non-stationary and insufficient data.
- Emphasised the importance of ample data and the effectiveness of simpler, classic statistical methods.

## FUTURE STEPS

- Explore alternative loss function, and input window, for ES-RNN to mitigate bias in predictions.
- Train the ES-RNN with all the series to exploit shared parameters and learn common local trends among the series.
- Address data non-stationarity and volume issues to enhance the performance of complex models.

-  S. Makridakis, E. Spiliotis, and V. Assimakopoulos.  
**The m4 competition: 100,000 time series and 61 forecasting methods.**  
*Int. J. Forecasting*, 36(1):54–74, 2020.
-  S. Smyl, J. Ranganathan, and A. Pasqua.  
**M4 forecasting competition: Introducing a new hybrid es-rnn model, 2023.**  
Accessed 15 Sep 2023.

-  A. Wibawa, A. Utama, H. Elmunsyah, et al.  
**Time-series analysis with smoothed convolutional neural network.**  
*J Big Data*, 9:44, 2022.
-  P. R. Winters.  
**Forecasting sales by exponentially weighted moving averages.**  
*Management science*, 6(3):324–342, 1960.