Project of Advanced Machine Learning : Neural Networks Fail to Learn Periodic Functions and How to Fix It

Virgile Batto VIRGILE.BATTO@ENS-PARIS-SACLAY.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Émilien Marolleau EMILIEN.MAROLLEAU@ENS-RENNES.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

Doriand Petit DORIAND.PETIT@TELECOM-PARIS.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

Émile Siboulet EMILE.SIBOULET@ENS-PARIS-SACLAY.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Editor: Machine Learning Avancé (2021-2022)

Abstract

Learning periodic or pseudo-periodic functions that will keep interesting behaviors outside the learning domains is impossible by classical activation functions such as relu or tanh. To solve this problem we can suppose that using non-linear periodic or pseudo-periodic functions would solve the problem. Moreover, the article is supposed to retrieve some classical ability for activation function such as classification task. This is not that simple, indeed multiple properties are needed to match the specifications of an activation function. We will try to reproduce the results obtained in Liu et al. (2020) which proposes to use the snake function as an activation function in the deep layers. To do so, we will compare the results of the article by varying the size of the different proposed networks and by comparing the results with other methods allowing to learn periodic or pseudo-periodic evolutions. The results found as a result of this work are less consistent than what was found in the original paper.

Keywords: Pseudo-periodic Functions, Machine Learning , Snake Funciton, Forcasting, Long Short-term Memory

Introduction

This work is based on the article Liu et al. (2020) which proposes a method of learning periodic functions to obtain predictions outside the learning range. Following this article, this function will

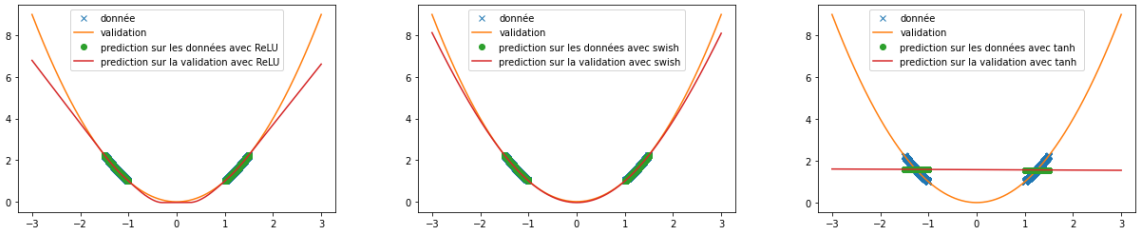
be implemented from this article in Tensorflow and Pytorch. This article is still under revision and has received favorable opinions with reservations on the reproducibility of results [1].

1. Testing of different activation functions on simple data

1.1 The classic activation functions

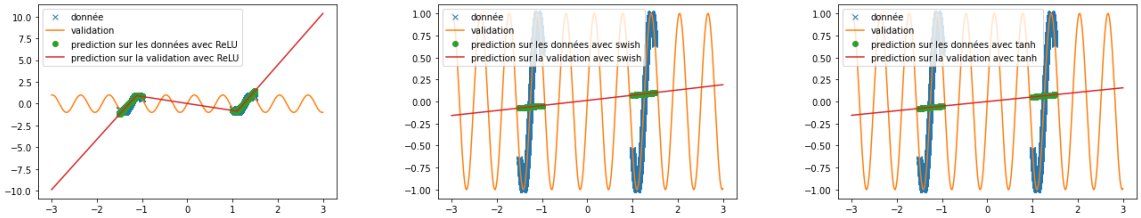
First, we set up classical activation functions (ReLU, swish and tanh) to predict semi-periodic or periodic signals. To do this we placed ourselves in the same condition as in the article "Neural Networkks Fail to Learn Periodic Function and How to Fix It", i.e. with a single hidden layer neural network of 512 neurons.

Different functions have been used to verify the ability of the ReLU, tanh and swish functions to predict signals. Here we are particularly interested in the prediction of the sine function and an x^2 function.



(a) ReLU (b) swish (c) tanh

Figure 1: Prediction of a square wave signal with different activation functions



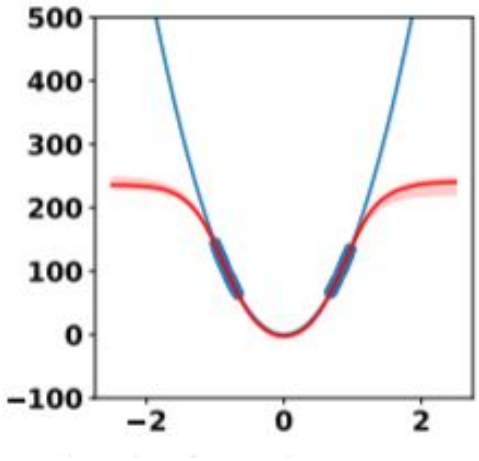
(a) ReLU (b) swish (c) tanh

Figure 2: Prediction of a sine signal with different activation functions

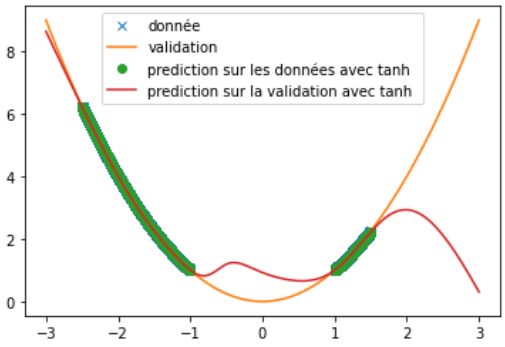
Thus, we can see that when predicting a square function, tanh gives the mean value of the point cloud while ReLU and swish provide a more accurate approximation (cf fig 1) . In addition, the small number of training data taken (in order to be as close as possible to the article) leads to a rather long convergence of the results. Here the training was stopped when the losses stagnated (after about 100 iterations) but it is possible that with a longer training the prediction becomes much better. This may explain the different results of the article since in this one, the activation

1. `https://openreview.net/forum?id=ysFCiXtCOj`

function tanH gives a more convincing result (cf fig 3a). A similar result is obtained when using a non-symmetric training base (see fig 3b)



(a) Prediction of Tanh presented by the article



(b) Tanh prediction with asymmetric BDD

However, when predicting a sine, the three functions do not provide relevant results at all (see fig 2). This is due to the small number of points given for training, which prevents the networks from overlearning and tries to force them to make a true prediction.

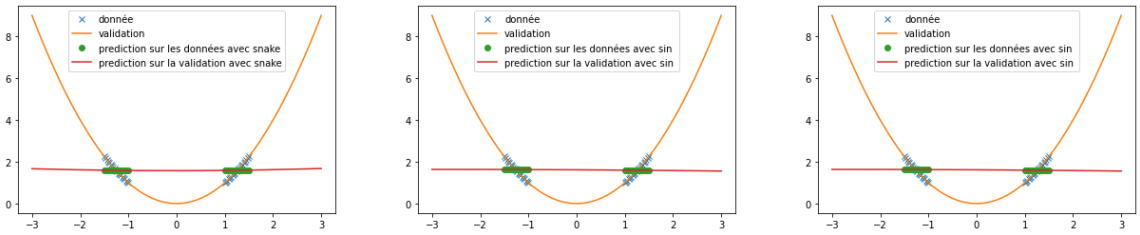
Thus, we can see very clearly that in the case of periodic function prediction, the classical activation functions are absolutely irrelevant. We are therefore interested in other activation functions.

1.2 New activation functions

A new activation function is implemented:

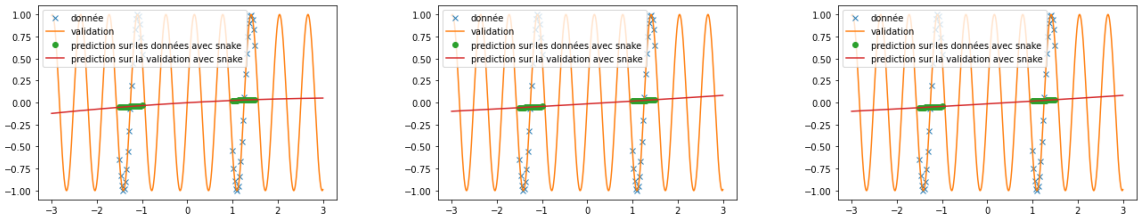
- the function $x + fracsin(x)$.
- the function $x + fracsin(x)$.
- the function $x + fracsin(x)^2\alpha$

we try it as before on the prediction of sine and the prediction of x^2. We use the same conditions as in the article, i.e. a single hidden layer network of 512 neurons and $\alpha = 10$. We obtain the following results:



(a) snake(x) (b) $x + \sin(x)$ (c) $\sin(x)$

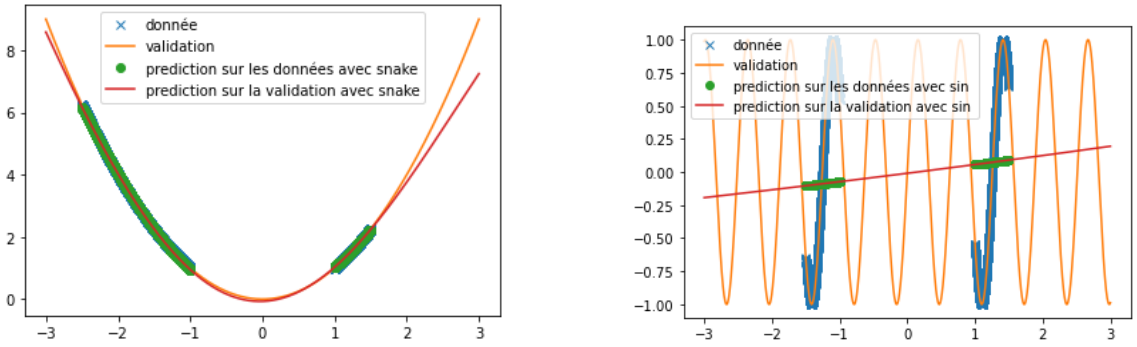
Figure 4: Prediction of a square with different activation functions



(a) snake(x) (b) $x + \sin(x)$ (c) $\sin(x)$

Figure 5: Prediction of a sine signal with different activation functions

We can see that, with training bases similar to the one presented in the article and an identical architecture, we do not obtain any coherent result. This is surely due to an extremely long convergence time which means that we never reach a good prediction. The results provided in the article are much more encouraging (cf fig: 10). Nevertheless, when using a training base with a higher density of points, we obtain much better results with the snake function, although these are still not relevant for the prediction of a sine:



(a) prediction with snake as activation (b) prediction with snake as activation

1.3 Attempt to make the new activation function work with other network architecture

Keeping the same form of training base as in the article (see the points in the figure 10), we change the neural network architectures in order to check if a correct prediction is still possible. We are interested in a classical sine function and we try to predict a sine signal of a different frequency.

Several tracks are explored:

- the decrease of the number of neurons
- the increase in the number of layers

in a first time we strongly decrease the number of neurons, we only put 4 hidden neurons:

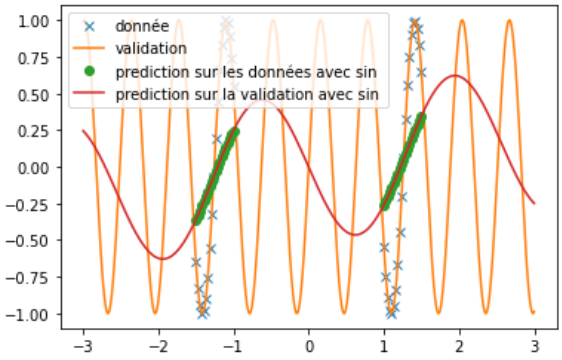


Figure 7: prediction d'un sinus par

The results are rather interesting and show that it would be necessary to have a number of data much higher than the number of neurons.

To solve the frequency problem, we made a network with 5 chained layers of 64 neurons and in order to virtually increase the number of data, we decrease the batch size while increasing the number of iterations. We obtain a rather conclusive result;

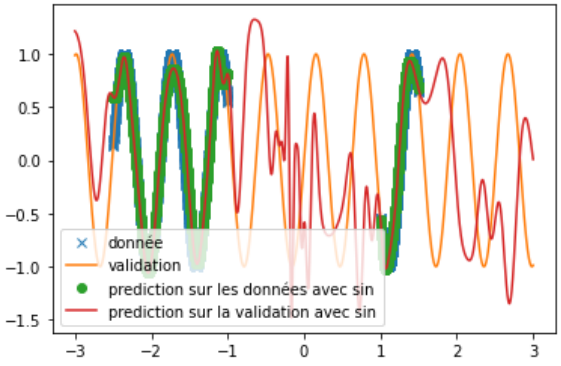


Figure 8: prediction with a hidden multi-layer network

We extrapolate these results to the snake activation function, and we use the same architecture, but with more training data and we obtain a result similar to the one presented in the article:

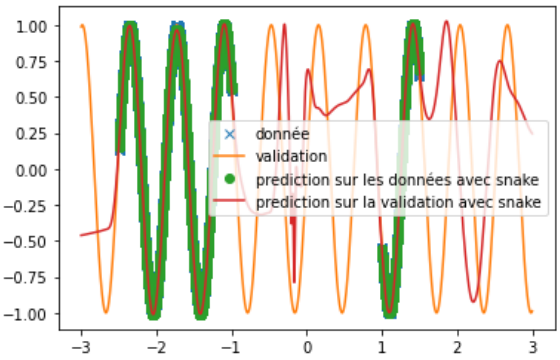


Figure 9: prediction with a hidden multi-layer network with snake in activation function

Thus, we obtain the same results as those presented in the article, but using extremely different architecture than those presented in order to overcome the convergence time problem.

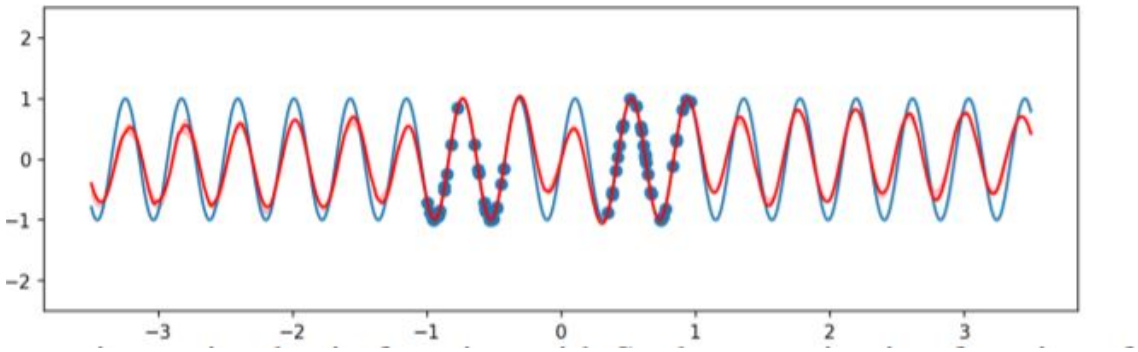


Figure 10: Prediction of a sine with the snake function, from the article studied

2. Snake / ReLu comparison on the Ciphar-10 training base

2.1 Problem setting

This article aims at showing that snake is a reliable alternative to activation functions, so it must be proved that it is as accurate as the others in common machine learning tasks. Indeed the main interest of the snake activation function is to be efficient with periodic and aperiodic functions. Where the other parts of this report focus on the question of prediction on periodic functions, this part allows to find the results of the article announcing that snake competes with other well-known activation functions, like ReLU. Thus, in accordance with the experiments of the article, a classification task has been reproduced on the Ciphar-10 training base.

2.2 Technical solution and implementation

According to the article, Snake manages to reach an accuracy of 93%. It is comparable to those obtained with other activation functions, especially in classification tasks. Thus to confirm this and in accordance with the article, the classification of images from the Ciphar-10 database has been trained using the convolution network ResNet-18. The representation of its architecture is shown on the figure 11. This one is made of convolution layers with a kernel of size 3x3, the size of the convolutions increases with the depth of the layers. Between each layer we add information from the previous layer, tht's why tis architecture is called ResNet (for Residual Network). The code implementing this network is made of 2 python scripts, the first one is object oriented and defines the ResBlock class which composes all the residual convolution network architecture, it also implements a class inheriting from ResBlock which defines the ResNet18 architecture. The second script calls all the tensorflow functions as well as the Ciphar-10 database and the ResNet18 class.

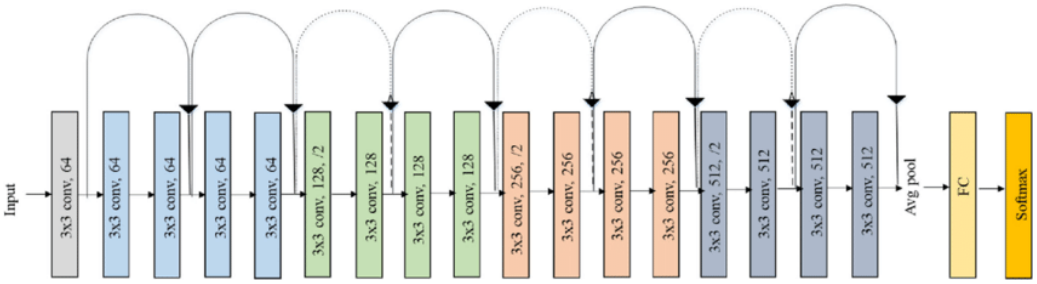


Figure 11: Architecture of the ResNet-18 residual convolutional neural network

2.3 Results obtained and comparisons with the article

The training phase was performed with a batch size of 256, the database is composed of 50,000 images, cut into 60% for training elements, 20% for validation data and 20% with test data. As in the article the data were trained on 100 epochs. The computations were run on an Nvidia RTX 3070 graphics card, for a computation time of 16 minutes. In order to compare the performances with a reference activation function the same calculations were performed with the ReLu function too. The results are visible on the figure 12 and on the figure 13.

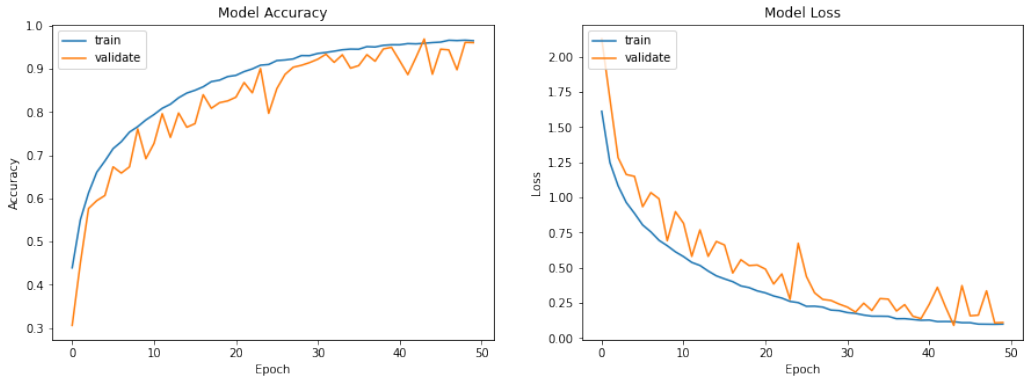


Figure 12: Prédiction avec ReLu comme fonction d'activation sur la base d'entraînement Ciphar-10

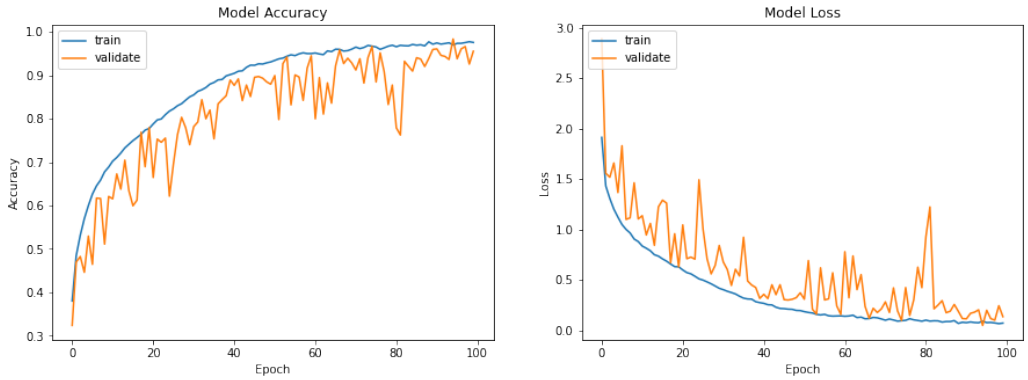


Figure 13: Prediction with Snake (a = 1) as activation function on the Ciphar-10 training base

As can be seen in the figures below, the training of the network with the snake activation function for 100 epochs reaches 0.9554 where the training with the ReLu function reaches 0.9611 in 50 epochs. We can therefore confirm the ability of Snake to reach the same performance in concrete classification tasks. However, Snake learns more slowly than ReLu. Indeed it took Snake twice as many epochs as Relu to reach the same results. However, the experiments performed here are therefore in good agreement with the results of the article.

3. Financial Data Prediction

3.1 Problem Setting

Hence, trying to predict periodic and quasi-periodic functions is not always an easy feat and the use of the Snake activation function can thus be a great help in resolving this issue. Following this idea, it is interesting to focus on another environment : the global economy. Indeed, the total US market capitalization has a quasi-periodic behavior (whether at a microscopic level or a macroscopic one) and the authors thus decided to analyse the snake effect on its predictions.

3.2 Data and Pre-processing

In order to measure the global economy, the authors decided to use the Wilshire 5000 Total Market Full Cap Index, a common financial index that gives daily samples (on working days exclusively). The dataset is clear and easy to obtain and we thus had no problem in copying the beginning and ending dates of each training and test sets. However, there were some NA values, which come from bank holidays. Rather than dropping them, the first choice was to interpolate their values. However, the paper talked about "around 6300 [training] points in total", while the interpolation gave 6544 points, hence we then decided to remove these interpolations to obtain a satisfying number of points. The only other pre-processing we made was some normalization over the y values, as well as over the x values (which are initially a list from 0 to the number of training points).

3.3 Technical Choices

According to the paper, we developed a feedforward neural network with 4 hidden layers (1 neuron - 64 neurons - 64 neurons - 1 neurons). The paper specifies the test of several activation functions (ReLU, Swish, sin + cos, sin and Snake). However, it does not precise which layers should have that varying activation function, and after some experiments, it was quite clear that the last layer did not have this special activation (the results with snake chosen for each layer were really bad, while removing the last activation function worked wonders).

We also chose arbitrarily an SGD optimizer with a learning rate of 0.001 and a momentum of 0.8.

Finally, the last choice to be made was about the parameter a of the Snake function. We've noticed an important variation of performances depending on its value, and thus decided to consider this decision as part of the results.

3.4 Results and Discussion

First of all, we have tried to test which snake parameter gave the best results (figure 14). In fact, this choice was dependent on our wish to have a high or a low "frequency" of the resulting quasiperiodic signal. Hence, we have noticed that the choice of the parameter a was crucial in the learning and the prediction of the funciton. A small value of a gave a very low frequency and thus a really important under-fitting. On the other hand, a high a was too difficult to learn as we can see with a value of 100. Finally, for in-between values of a, the choice is ours to make, as we can choose the "degree" of precision we want (do we want to predict the general trend only or do we also want to predict higher-frequency changes). For the rest of this part, we have decided to work with a = 30.

Method	MSE on test set - Our version	MSE on test set - Paper's version	Training Loss
ARIMA(2, 1, 1)	0.01605	0.0215 +- 0.0075	-
ARIMA(2, 2, 1)	0.01614	0.0306 +- 0.0185	-
ARIMA(3, 1, 1)	0.01604	0.0282 +- 0.0167	-
ARIMA(2, 1, 2)	0.01602	0.0267 +- 0.0154	-
ReLU DNN	0.1205 +- 0.0886	0.0113 +- 0.0002	0.0330
Swish DNN	0.0119 +- 0.0012	0.0161 +- 0.0007	0.0012
Sin + Cos DNN	0.0125 +- 0.0019	0.0661 +- 0.0936	0.0017
sin DNN	0.0118 +- 0.0029	0.0236 +- 0.0020	0.0019
Snake	**0.0091 +- 0.0012**	**0.0089 +- 0.0002**	Vanishing

Table 1: Predictions of Wilshire5000 during the test set

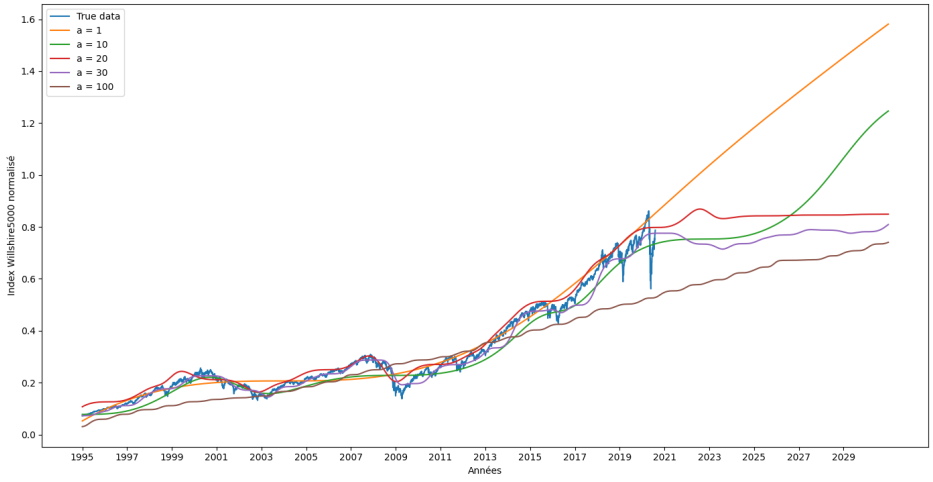


Figure 14: Predictions of Wilshire5000 for various values of a

Then, the following goal was to obtain a similar figure and table of the paper with test mean square errors of DNNs using different activation functions, as well as more classical financial models named ARIMA (tab. 1). Indeed, we were glad to obtain similar trends and magnitudes of result.

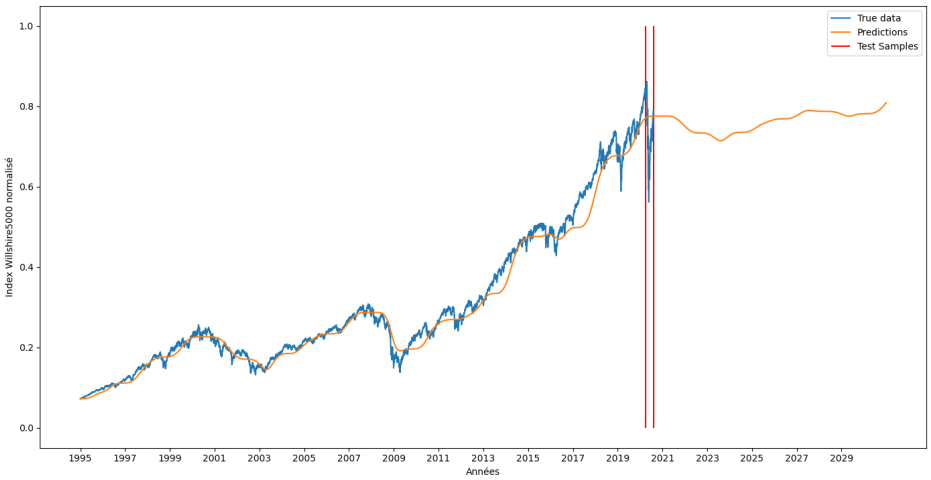


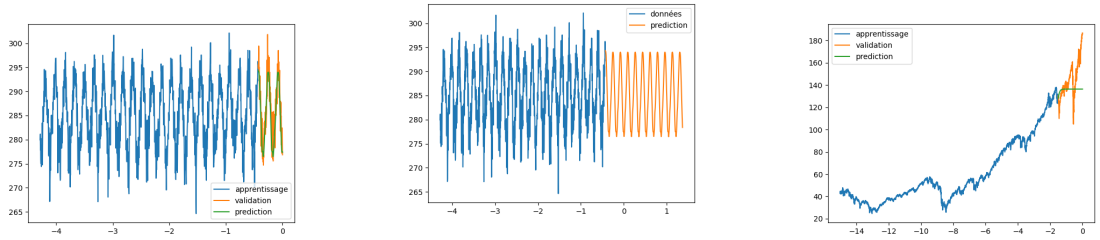
Figure 15: Predictions of Wilshire5000 for a = 30

However, we still had some differences with the authors' results. First of all, the ARIMA computations gave quite different results, as their method was left unknown. Ours is deterministic (hence no standard deviation), and the different choices of orders also resulted in similar MSE while the authors had way more various errors. Moreover, we can also notice that, while most of the activation functions results in similar MSE, our Relu DNN has a 10 times higher error. This is quite surprising, and the reason is uncertain. As explained in the next paragraph, the test set's choice might be part of the problem.

Please note that, while the Snake activation function presented the best results whether in the paper or in our own tests, the author's choice defining the test set is at the very least questionable, if not straight up a bad choice. Indeed, these four months contain the "black thursday", a sudden market crash that cannot be predicted from the training samples, and that was way too quick for our estimated function to follow. Thus, comparing the MSEs on this test set might not be a pertinent indicator. A possible solution would be to modify the test set, but the goal of this project was to obtain the same results as the paper's ones rather than prove the efficiency of their idea. We can nonetheless compare the training loss as it may bring some more informations and we directly notice that, as the authors mentionned, the Snake neural network is the only model which trains to vanishing training loss, only it can learn precisely the entire training range of the indicator.

4. Use of LSTM layers

The most common way to do periodic and pseudoperiodic function prediction is to use LSTMs. This part is only very little developed in the article, however we decided to develop it a little more. This approach allows us to contrast the different results found for prediction. We will approach in



(a) Weather forecasting

(b) Verification of the temporal staibility

(c) Wilshire5000

Figure 16: Prediction by networks using LSTM layers

particular the case of the Wilshire5000 because the results found by the method of the article are very debatable in view of its evolution (Sunder (2021)).

4.1 Weather forecasting

We will try to predict the average temperature per week of the observatory of Clermont-Ferrand [2]. To do this, we give ourselves a series of measurements that we group together by batch so that the neural network is able to predict the next measurement in a sequence. By the capacity of the LSTM layers to be stable by composition one will be able, with a trained network, to apply successively the network to predict the temperatures on a long temporal range (Figure 16a).

We notice errors relative to the temperature amplitude of the order of 200%. This is important but can be explained by the high unpredictability of temperatures. Moreover the fact of taking into account only the data of the station makes the prediction very little precise. A way to treat the problem more globally would be to use convolution layers taking into account the geographical distribution of the different measurements Agrawal et al. (2019).

We also verify the stability in time and the extrapolation of the prediction. We notice that the composition of LSTM is very stable. We can also conclude that there was exploitable knowledge in the weather data. As a matter of fact, we find an oscillatory behavior in a model which does not include a periodic function.

4.2 Wilshire5000

In this part we will use the same method as in the previous part. Indeed, we are going to constitute sequences from the measurements of the Wilshire5000[3] in the hope to predict the following value and to apply it to predict longer sequences.

We notice that we did not succeed in finding a knowledge in the measurements of wilshire5000(Figure 16c). Indeed the solution found by the neural network is to give the value at the previous time step. Moreover, by comparing the evolution of this indicator with what actually happened afterwards (Sunder (2021)), we see that the prediction was quite wrong. The snake

2. `https://donneespubliques.meteofrance.fr/?fond=produit&id_produit=90&id_rubrique=32`
3. urlhttps://fred.stlouisfed.org/series/WILL5000INDFC

activation functions as well as the LSTMs are not able to predict the evolution of an indicator of the stock market only by its previous evolution.

Conclusion

We have succeeded in reproducing some of the results of the article. First, we succeeded in showing empirically that the snake function could be used as an activation function. Then, we showed that it is generally less efficient than the unavoidable relu function but it allows to learn non-trivial databases such as Ciphar-10.
Nevertheless, we observed that the parameter a in the snake activation function has a great influence on the prediction. This is a major issue because its arbitrary parameterization strongly impacts the prediction and can lead to false predictions on stock market indicators for instance. LSTM networks are more robust in reproducing periodic and pseudo-periodic functions while remaining stable over time.

References

Shreya Agrawal, Luke Barrington, Carla Bromberg, John Burge, Cenk Gazen, and Jason Hickey. Machine learning for precipitation nowcasting from radar images, 2019.

Ziyin Liu, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. *CoRR*, abs/2006.08195, 2020. URL `https://arxiv.org/abs/2006.08195`.

Shyam Sunder. How did the u.s. stock market recover from the covid-19 contagion? *Mind & Society*, 20(2):261–263, Nov 2021. ISSN 1860-1839. doi: 10.1007/s11299-020-00238-0. URL `https://doi.org/10.1007/s11299-020-00238-0`.