Time Series Prediction via Convolutional Neural Network

Amir Shirian, *Student Member, IEEE*, Ahmad Kalhor, *Member*, IEEE, B.N.Araabi, *Member*, IEEE

***Abstract*— Prediction or finding ways to have correct estimation of system's behavior, have been attended since beginning in science, engineering, economy and so on. Classic methods often depend on hand-crafted features that were expensive to create and required expert knowledge of the field. In other way, convolutional neural networks (CNN) have showed success in achieving adequate features automatically and the lowest error rate in different tasks. In this paper, we proposed an injective mapping based on normalized fast Fourier transform (FFT) to convert time series into two dimensional array and used columnar CNN with a novel lost function in order to extract essential features automatically. In order to examine succession of this method in comparison with conventional methods, we used Mackey Glass and Sun spot time series as standard benchmarks and gold price as real world time series.**

***Index Terms*— Convolutional Neural Network, Injective Map, Fast Fourier Transform, Gold Price Prediction, Time Series Prediction.**

I. Introduction

TIME series is one of the most important topics in scientific and financial applications. One of the main issues of science is prediction and find out how one signal perform over future time. When we have information about future, we can easily deal with and have a best respond to gain maximum profits. Prediction of time series was researched in science, economy and even industry and have had great budget since today. Over the years, building an explanatory model and finding its parameters which the model have the best fit over time, was one of the popular approaches in these problems. Auto-regressive (AR) and AR integrated moving average (ARIMA) models are one of the conventional statistical methods which used weighted sum of past values to estimate future values. In 1983, Newbold proposed a method to analyze and forecast based on the construction of ARIMA model and become one of the earliest paper in these field [1]. These methods assumed statistical relation between lags of time series which are not always true. In 1996, Masumi et al. illustrated that: "A central issue common to all of them is the determination of model structure" and proposed three neural network models and measurements to select model efficiently [2]. In 1998, Vapnik and his colleagues introduced Support Vector Machines (SVM) that used risk function consist of the empirical error and a regularized term which is derived from the structural risk minimization principle [3]. In 2003, Kyoung-jae Kim applied SVM theory on financial forecasting problem and compared results with back-propagation neural networks and case-based reasoning [4]. SVMs have a regularization term which avoids overfitting and become results more robust and generalized. Furthermore, SVM uses quadratic programming and don't trap on local minima. However, they have some of disadvantages which can't solve all machine learning problems. SVMs must formulate problem as two class classification and take long time in both training and testing procedure.

An artificial neural network (ANN) can be useful for nonlinear mapping between input and output domain. A typical ANN filtered input layer with one or more hidden layer which each consist of nodes, before they reach the output. ANNs with their remarkable ability to derive meaning from complicated data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Multi-Layer Perceptron (MLP) network is one of the most used ANNs in time series prediction. MLP have showed success in classification and regression problems. In 2010, Christophe Paoli et al. proposed an ad hoc time series pre-processing method before MLP and compare with ARIMA (classical method), Bayesian inference, Markov chain and KNN predictors [5].

In 1997, Long Short-Term Memory (LSTM) was proposed as a RNN architecture and had been shown to outperform traditional RNNs on numerous processing tasks [6-9]. Gers et al. used LSTM with various size of input window and compared results with other ANN [10]. In 2015, YongxueTian et al. said: "Most of the models require the length of the input historical data to be predefined and static, which cannot automatically determine the optimal time lags." They proposed a new LSTM based model to overcome this shortage [11]. Also another extensions of RNN were proposed and used as wavelet neural network (WNN) in mapping nonlinear functions and diagonal recurrent wavelet neural network (DRWNN) [12].

Another approach to predict time series is to combine fuzzy logic and neural network to perform a model and predict. In 2002, Nikola K. Kasabov et al. introduced a new type of fuzzy inference systems, denoted as dynamic evolving neural-fuzzy inference system (DENFIS), for adaptive online and offline learning, and their application for dynamic time series prediction [13]. In 2008, WojciechStach et al. used fuzzy cognitive maps (FCMs) and a learning method that uses real-coded genetic algorithm [14].

In recent years, many new approaches have been introduced. In 2016, Elena Mocanu et al. investigated two newly developed stochastic models for time series prediction of energy consumption, namely Conditional Restricted Boltzmann

Machine (CRBM) and Factored Conditional Restricted Boltzmann Machine (FCRBM) [15].

A real-world time series conveys encoded information from various hidden sources; thus, using methods which try to extract input effective features directly from the time series in its one dimensional frequency or time domain form, is not sufficient. Representing a time series as a two dimensional signal supposed to be better solution to decode information and extract effective features for prediction purposes. Actually there are some methods in representing a signal as two dimensional forms. Petrosian et al. used wavelet decomposition to extract specific signal features and combined with recurrent neural networks (RNN) to predict EEG signal [16]. In 2013, Osamma et al. explored multiple aspects of CNNs in speech recognition problem [17].In 2014, Ossama et al. and his colleagues matured their idea and used CNN to solve speech recognition problem. They used spectrogram , first and second temporal derivatives as a RGB image [18]. In contrast, during recent decade, deep neural networks have been developed as powerful tools which could extract features from two-dimensional signals. Convolutional Neural Networks (CNN) have led to breakthrough results on a variety of pattern recognition problems, such as computer vision [19], voice recognition, object recognition [20, 21].

Here, it is desired to represent a time series as a two-dimensional temporal signal by using FFT technique then using CNN to achieve more effective features in prediction of time series. The rest of the paper is organized as follows: Section II describes CNNs and their applications. Section III describes how the present paper's concept was applied to prediction problem. Section IV describes how we prepare data and programming language which we used. Finally, section V concludes the paper.

II. CONVOLUTIONAL NEURAL NETWORK

In last few years, Convolutional Neural Networks (CNN) have successfully been applied to analyzing visual imagery. From Hubble and Wiesel's work on monkey's visual cortex [22], we understood that the visual cortex contains hierarchical cells. Each group of cells are sensitive to small sub-region of visual field that called receptive field. These cell groups are lied alongside to support all visual field. CNNs were inspired from visual cortex and they are biologically extensions of MLPs. MLP is a hierarchical fully connected network but in CNNs, the input of each neuron in layer n are from a subset of neurons (receptive field) in n-1 layer that we know as sparse connectivity. In Fig. 1, imagine that the layer n-1 in the input, neurons in layer n have receptive fields of width 3 and this means each neuron only connected to 3 adjacent neuron in previous layer. This architecture cause in learning step that each receptive field become an expert exactly on same sub-region. Hubble and Wiesel, also introduced two kind of cells as simple and complex cells. Simple cell responds to oriented edges and edge-like patterns in fixed receptive field but complex cell responds to them in a degree of spatial invariance and in larger receptive fields. They used these biological structures idea to create convolutional and subsampling layers.

CNNs had been developed with four ideas: local receptive fields, shared weights, spatial subsampling and the use of many layers. One of the profits of this network is shared weights that reduces number of parameters.

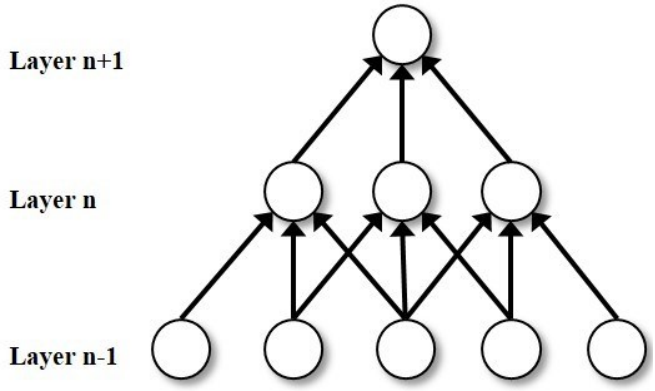

Fig. 1. Sparse connectivity in convolutional neural networks

A typical CNN is shown in The network consist of three kind of layers: convolutional layer, subsampling layer and fully connected layer [23]. Each convolutional layer has one or more local window which is scanned over the whole plane to extract appropriate features and subsampling layer uses to reduce size of each plane. A typical convolutional layer is shown in Fig. 2. Fully connected layers are convenient ones that uses for classify extracted features form previous layers. CNNs have been successfully applied to image classification from beginning. In early 1990s, they used time-delay neural networks to recognize phonemes [24]. CNNs were also used as object detection in natural images including faces [25-27], segmentation of biological images [28], natural language processing [29], speech recognition [30]. In 2012, Hinton and their colleagues caused great attraction for computer vision and machine learning community to use CNN in ImageNet competition [21]. In early years, there is more interest in using CNNs in industry problem like autonomous mobile robots and self-driving cars [31, 32]. In this article, we used extension of convolutional neural network and used unknown ability of this network to extract feature from 2D arrays and use for prediction problem.

III. APPLYING CNN CONCEPT TO TIME SERIES

In prediction problem, we are trying to find time series distribution over time or frequency domain to extrapolate or estimate beyond the original observation range. A single time series only has time domain information and we couldn't decode information about long term and short term behaviors. In order to increase our information about data, we can analysis time series in time-frequency domain.

Convolutional Neural Networks have shown success in low rate error in classification tasks. This success is largely indebted to use of convolutional and max-pooling layers to extract unsupervised features. In this work we will show that we can use of these feature extractors to solve prediction problem.

In this Section, we are probed with adaptation of CNN principles to time series prediction problem. First of all, we

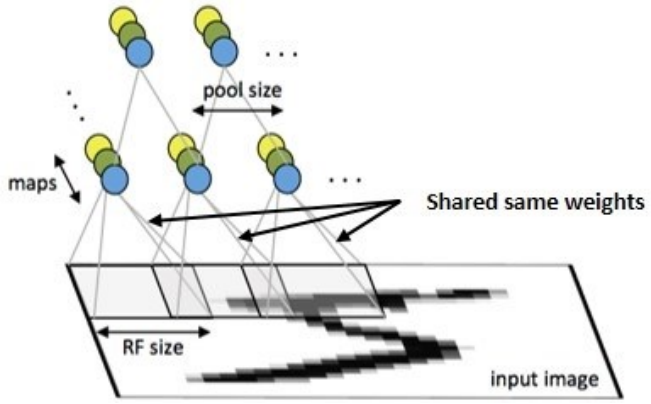

Fig. 2. First layer of a convolutional neural network with pooling. Units of the same color have tied weights and units of different color represent different filter maps.

know that CNNs are designed to process data that come in the form of multiple arrays for instance an image consist of three 2D arrays. So for applying time series data to the CNN, we must use a kind of injective mapping that consider two principles, it must expand the dimension in number and ability to indicate the value of time series along each column. In the following, we introduce a modified "Fast Fourier Transform" transform that satisfices our aims.

A. Injective Mapping

A Fast Fourier Transform (FFT) computes the discrete Fourier transform of a sequence. Fourier transform converts a time series from its original domain to a representation in the frequency domain. FFT is a one-to-one transform, which means each time series has specific features in frequency domain. Consider the discrete Fourier transform of a function $f[n]$:

$$F(j\omega) = \sum_{k=0}^{N-1} f[k] e^{-j\omega kt} \tag{1}$$

Where ω is frequency and N is the number of data points in a window, which this window sweep all time series. We could in principle evaluate this for any ω, but with only N data points to start with. In this work, we tried to evaluate the amplitude of sin and cos in each window and then kind of normalization to create the amplitude of each frequency.

$$\text{Image}_{r_{\text{column}}} = \frac{1}{N} \times (AmpC \times \cos(Wt) + AmpS \times \sin(Wt)) \tag{2}$$

Where N, W, AmpC, AmpS are number of frequencies, vector of frequencies, amplitudes of cos frequencies and amplitudes of sin frequencies. In this equation, all of operations are applied element-wise. This equation returns information about importance of each frequency in building of time series and this normalization cause summing up each column returns value of time series in each time samples.

Each fast Fourier transform has a window of lags that return the amplitude of each frequency in that period. Each frequency can illustrate a specific behavior of time series, for instance low frequencies show us that this time series has a long term behavior which its power has direct relation with its amplitude. This window moves over all time series and we have an amplitudes column for each window. We used this transformation to convert a time series into the 2D dimensional

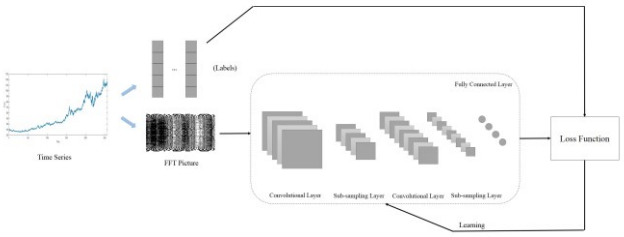

Fig. 3. An illustration of our method in this paper.

picture. Each columns of this picture have frequencies information that we use as an input of CNN. In order to create our dataset, we split this 2D long picture to many 2D pictures with constant length as mini pictures for input data and one column vector, exactly one column after each mini pictures as label that is shown in Fig. 4. In other words, we want to feed CNN with this mini pictures to learn their labels. One of the advantages of this transform is behavior prediction in long term. Each column consist of power of each frequencies over last window and we can predict which behavior (fast or slow in time) will be occur in future.

B. Loss Function

Traditional CNNs have showed success in achieving low rate error in task of classification and common cross entropy and maximum likelihood (ML) may be used to validate our error. In prediction task, they attempted to find a nearest distribution to real data distribution. However, learning ML cost function with gradient, given mini batch of training data is extremely noisy and has a high variance. Gradient methods need to sample from model distribution that is non stationary and the result often is sparse in high dimensional space.

In recent studies, Schuurmans and his colleagues tried to proposed reward augmented maximum likelihood to optimize a Kullback-Leibler divergence between the exponentiated reward and model distribution [33].

In traditional method, given a set of input-output pairs, $D \equiv \{(x^i, y^i)\}_{i=1}^N$, models learn parameters to amplify distribution of $p_\theta(y|x)$,

$$\theta^* = \arg\max_\theta p_\theta(y|x) \tag{3}$$

These objective functions are too hard for numerical optimization and often were used negative log-likelihood of parameters,

$$L_{ML}(\theta; D) = \sum_{(x,y)\in D} -\log p_\theta(y|x) \tag{4}$$

According to (3), all wrong outputs are equally wrong and none is preferred among others.

In order to differ between all outputs, reinforcement learning (RL) was used with a maximum entropy regularizer [34], which is formulated as minimization objective,

$$L_{RL}(\theta; \tau, D) = \sum_{(x,y)\in D} \Big\{ -\tau H(p_\theta(y|x)) - \sum_{y'\in Y} p_\theta(y|x) r(y'|y) \Big\} \tag{5}$$

Where $r(y'|y)$ denotes the reward function, τ denotes regularization term and $H(p_\theta(y|x))$ is the entropy of $p_\theta(y|x)$. Entropy is a measure of randomness in the

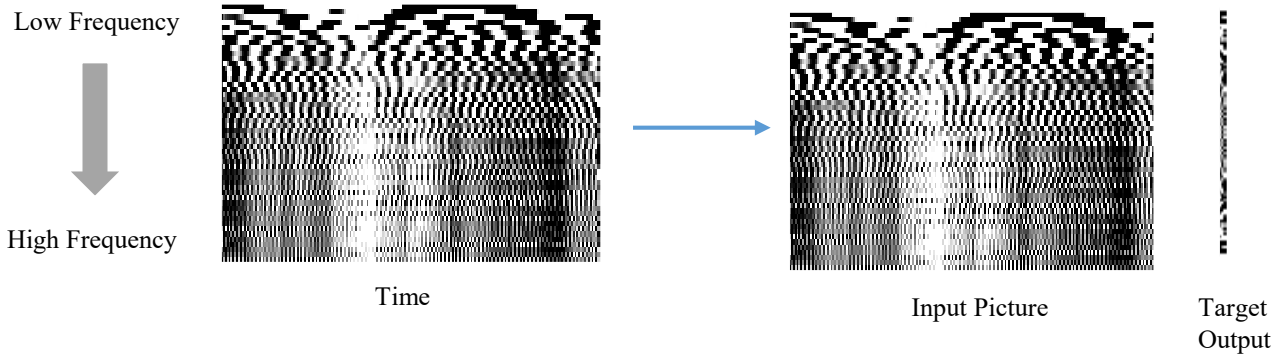

Fig. 4. FFT picture and how input and label extracted form.

information being processed. In other word, it shows uncertainty in appearing any output which is formulated as $H(p(y)) = -\sum_{y \in Y} p(y)\log(\mathrm{p}(y))$.

It is obvious that optimizing (5) using gradient is challenging cause of large variance of the gradients. Schuurmans, proposed an *exponentiated payoff distribution* to link ML and RL objectives:

$$q(y'|y;\tau) = \frac{\exp\{r(y,y')/\tau\}}{\sum_{y' \in Y}\exp\{r(y,y')/\tau\}} \quad (6)$$

In statistics, the Kullback–Leibler divergence is a measure of how one probability distribution diverges from a second expected probability distribution. For discrete probability distributions P and Q, the Kullback–Leibler divergence from Q to P is defined to be:

$$D_{KL}(P\|Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)} \quad (7)$$

In other words, it is the expectation of the logarithmic difference between the probabilities P and Q. In order to use ML advantages in learning procedures, RL objectives is expressed:

$$\sum_{(x,y)\in D} D_{KL}(p_\theta(y'|x)\|q(y'|y;\tau)) = \frac{1}{\tau}L_{RL}(\theta;\tau) + \sum_{(x,y)\in D}\log(\sum_{y' \in Y}\exp\{r(y,y')/\tau\}) \quad (8)$$

Where the second part in RHS is constant. (8) shows the minimum of $D_{KL}(p_\theta\|q)$ and L_{RL} is achieved when $p_\theta = q$. Schuurmans and his colleagues, proposed a novel method called *reward-augmented maximum likelihood (RAML)*, which generalizes ML by allowing a non-zero temperature parameter in the exponentiated payoff distribution, while still optimizing the KL divergence. The RAML objective function takes the form,

$$L_{RAML}(\theta;\tau,D) = \sum_{(x,y)\in D}\{-\sum_{y' \in Y} q(y'|y;\tau)\log(\mathrm{p}_\theta(y'|x))\} \quad (9)$$

Which can be re-expressed in term of a KL divergence as follows,

$$\sum_{(x,y)\in D} D_{KL}(q(y'|y;\tau)\|p_\theta(y'|x)) = L_{RAML}(\theta;\tau) - \sum_{(x,y)\in D} H(q(y'|y;\tau)) \quad (10)$$

In this paper, the objective function consists of the proposed RAML objective function which used to learn distribution over data and a term of weighted mean squared error to follow prediction value for each data point. The objective function takes the form:

$$L(y,y') = L_{RAML}(y,y') + \alpha\sum_i\left(y^{(i)} - y'^{(i)}\right)^2 \quad (11)$$

C. Model Structure

The CNN consists of one or more pairs of convolutional and max-pooling layers, where the bottom layers process a low length of each column independently to generate higher level representation with lower frequency resolution. In this model, we used Batch Normalization (Ioffe & Szegedy, 2015)[35] which stabilizes learnings by normalizing the input to each unit to have zero mean and unit variance. This helps with conquering the initialization problem and gradient flow in deeper models. For activation function, we used leaky Relu to avoid "dying Relu" problem. Instead of the Relu being zero when x < 0, a leaky Relu will instead have a small negative slope (of 0.01, or so).

$$f(x) = 1(x < 0)(\alpha x) + 1(x \geq 0)(x) \quad (12)$$

Where α is a small constant.

In training stage, CNN is estimated using standard Adam optimizer and RMSProp (Root Mean Square Propagation) algorithm to minimize defined loss function in III.B. All weights were initialized from a zero-centered Normal distribution with standard deviation reverse size of inputs.

IV. IMPLEMENTATION

A. Programming Language

Python's Tensorflow library [36] was used to implement the neural networks. It has many advantages: i) it will compile the functions using C and CUDA giving high performance, ii) Tensorflow has better graph visualizations than others, iii) it is better at second order of gradient because of its native support of symbolic computation, iv) its computational graphs can be distributed on a cluster for computations, v) it can use GPU's for further growth performance.

TABLE I
Prediction error rate of smooth monthly sunspot time series from 1749 to 2017

Model	RMSE	MAPE	Execution Time(sec)	Number of Layers
MLP	0.0074	2.925	437	4
RBF	0.0256	10.81	36	1
Multi-RNN	0.0154	4.7192	298	6
CNN (100 epoch)	2.4105e-4	0.88	4717	3
CNN (30 epoch)	2.5262e-4	0.92	1287	3

TABLE II
Prediction error rate of Mackey Glass time series with n=10, τ=20

Model	RMSE	MAPE	Execution Time(sec)	Number of Layers
MLP	0.0037	0.76	1356	4
RBF	0.0017	0.5031	107	1
Multi-RNN	5.7214e-4	0.37702	283	6
CNN (100 epoch)	7.2547e-6	0.05125	6052	3
CNN (30 epoch)	2.0191e-5	0.1263	1856	3

B. *Data Processing*

No pre-processing was applied to training images besides scaling to the range of [-1, 1] (standard normalization). All models were trained with mini-batches in size 30 with Adam and RMSProp optimizer with tuned parameters. The Leaky Relu is used as an activation function over the CNN and the sigmoid function over the MLP. In leaky Relu, the slope of the leak was set to 0.2 in all models.

C. *Machin Specifications*

All of the designed structures runs on a single PC with i7-6800K CPU @ 3.40GHz × 12, two parallel GeForce 1080 GPU and 64GB RAM.

V. EXPERIMENTS AND RESULTS

In this paper, we've tested three data benchmarks for evaluating prediction machine on the chaotic time series problem. Three state-of-the-art methods were employed to perform the comparisons: Multi-RNN, RBF and MLP.

We evaluate the performance of our model over training time and error measurements by comparison to other networks. The root mean squared error (RMSE) and mean absolute percentage error (MAPE) are used to measure the prediction performance.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$
$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad (7)$$

Where y_i, $\hat{y}_i$ and $\bar{y}_i$ are observed data, predicted and average of data.

A. *Sun Spots*

Sunspots are temporary phenomena on the photosphere of the sun that appear as dark spots compared with surrounding regions. They are areas of reduced surface temperature caused by concentrations of magnetic field flux that inhibit convection. Recent studies are shown that sun spots have chaotic behavior. We trained proposed models on sunspot time series and the results are shown in TABLE I. In this table, our method has a very smaller prediction cost against others but has a longer training time than others. Fig. 5 shows the smoothed monthly WDC-SILSO sunspot and predicted output model number over 268 years totaling 3215 records.

B. *Mackey Glass*

The Mackey Glass equation have had a great impact on mathematical studies of delay-differential equations. Mackey Glass equation was used in physiological science and was applied to model diseases [37]. Mackey-Glass equation is the nonlinear time delay differential equation which depending on the values of the parameters, this equation displays a range of periodic and chaotic dynamics.

$$\frac{dx}{dt} = \beta\frac{x_r}{1+x_r^n} - \gamma x \quad (6)$$

Where β, γ and n are real numbers and x_r represent the value of the variable x at time $(t-\tau)$. We solved this equation with $\beta = 0.25, \gamma = 0.1, n = 10$ and $\tau = 20$.

TABLE II shows training results over four different models. As before, it shows that our proposed method outperforms prediction cost but has a longer trainig time in comparision with other models. This is supported by Fig. 6 which shows the difference between predicted time series and original.

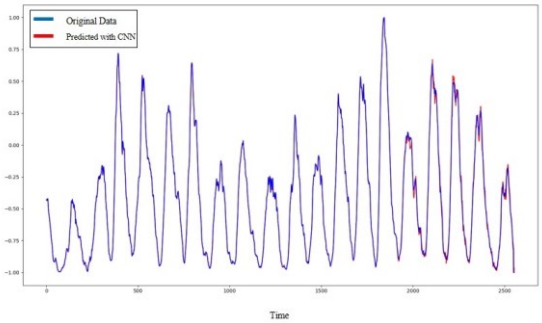

Fig. 5. Smoothed monthly WDC-SILSO sunspot number over 268 years totaling 3215 records (blue) and predicted values with CNN (red).

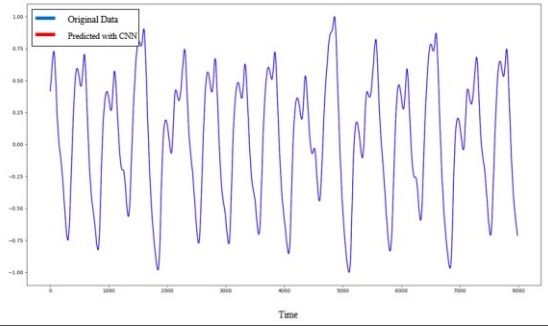

Fig. 6. Mackey Glass time series with n=10, =20 (blue) and predicted values with CNN (red).

TABLE III
Prediction error rate of gold price over 7 years.

Model	RMSE	MAPE	Execution Time(sec)	Number of Layers
MLP	0.0763	5.9374	1587	5
RBF	0.3088	8.1072	33	1
Multi-RNN	0.0556	29.36	79	6
CNN (100 epoch)	0.0078	3.6731	914	6
CNN (30 epoch)	0.0763	5.9374	1587	5

C. Gold

Prices values and stocks are one of the most interested field in economics and of science interest in financial time series prediction. In order to evaluate our model on real world time series, we used daily gold price time series over 7 years.
Always there was a challenge to predict marketing time series, so we studied more detailed on gold price's results. In Fig. 7, we can see loss function decay with two optimization approaches. In training procedure, we select learning rate higher than normal that can evade from local optimums. So far, in Fig. 7, we can see some growth in training loss but it would decay in time to prohibit this behavior in passing of time.
The comparative predicted results are shown in Fig. 8. All models have same FFT window, same set of training data, tuned and optimized parameters to obtain best results. This results are supported by TABLE III which shows prediction error and training time for four proposed model. This result also showed that our CNN method work well on real world time series.

D. Number of Layers

In this part, we investigate importance of structure. In our proposed CNN, one of the hyper-parameters is number of layers and we trained our model with sun spot data in 3, 4,5,6,9 and 12 layers with constant filter depth and size.
The results are supported by Fig. 9 which shows that more layers doesn't support descending in prediction error. In big networks, there are more parameters that needed to be updated and this cause that with equal input data, the over parameter network cannot emerge to best minimum in parameters space.

E. Activation Functions

In biologically inspired neural networks, the activation function is usually an abstraction representing the rate of action potential

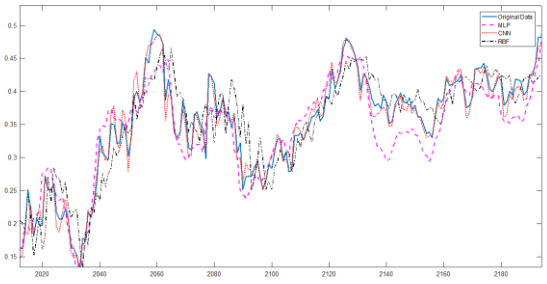

Fig. 8. Comparative results (prediction outputs on test data) showing the improvement brought by using CNN model.

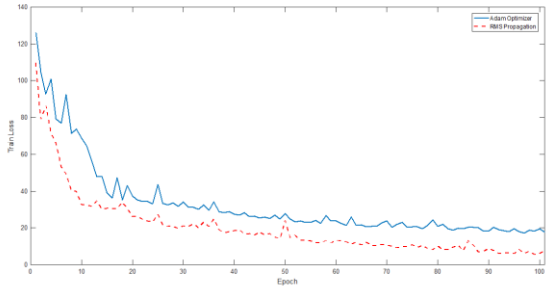

Fig. 7. Adam optimizer and RMS propagation loss in training process. In optimization process, learning rate was selected higher than normal that avoids trapping in local minima.

TABLE IV
Prediction error over different activation functions

Gate Function	RMSE	MAPE
Sigmoid	6.7934e-3	1.73
hyperbolic tangent	6.3942e-3	1.58
Relu	1.2873e-3	1.02
Leaky Relu	2.5262e-4	0.92

firing in the cell. In this part, we used different gate functions for output of each layer and investigated their results.
We trained sun spot data with sigmoid, hyperbolic tangent, Relu and Leaky Relu activation function. This results are supported by TABLE IV which shows that Leaky Relu has best prediction error in comparison with other activation functions.

VI. Conclusion

In this paper, we have proposed a method to convert one dimensional time series into two dimensional array and it has been shown that CNN can use on time series prediction problem. From the previous sections, it appeared that CNN outperformed both benchmark and real world time series. We used two standard benchmark to evaluate our proposed model and as we expected, it was outperformed. Also we trained our CNN model on real world time series and showed that can predict with lowest error rate in comparison with well-known methods for prediction problem. However, if no specific computational approaches are used, a disadvantage of convolutional neural networks is many calculations and much time required to find optimal solution.
Further direction for research include: (1) improvement of CNN structure for a better learning; and (2) use other transformation of time series and feed them to CNN as RGB input.

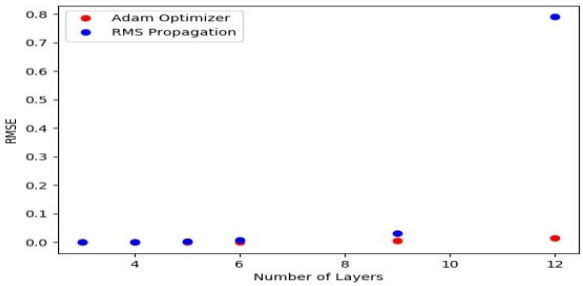

Fig. 9. Prediction error over different number of layers.

References

1. Newbold, P., *ARIMA model building and the time series analysis approach to forecasting.* Journal of Forecasting, 1983. **2**(1): p. 23-35.
2. Ishikawa, M. and T. Moriyama, *Prediction of time series by a structural learning of neural networks.* Fuzzy Sets and Systems, 1996. **82**(2): p. 167-176.
3. Vapnik, V.N. and V. Vapnik, *Statistical learning theory*. Vol. 1. 1998: Wiley New York.
4. Kim, K.-j., *Financial time series forecasting using support vector machines.* Neurocomputing, 2003. **55**(1): p. 307-319.
5. Paoli, C., et al., *Forecasting of preprocessed daily solar radiation time series using neural networks.* Solar Energy, 2010. **84**(12): p. 2146-2160.
6. Gers, F.A. and E. Schmidhuber, *LSTM recurrent networks learn simple context-free and context-sensitive languages.* IEEE Transactions on Neural Networks, 2001. **12**(6): p. 1333-1340.
7. Gers, F.A. and J. Schmidhuber. *Recurrent nets that time and count*. in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*. 2000. IEEE.
8. Gers, F.A., J. Schmidhuber, and F. Cummins, *Learning to forget: Continual prediction with LSTM.* 1999.
9. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
10. Gers, F.A., D. Eck, and J. Schmidhuber, *Applying LSTM to time series predictable through time-window approaches*, in *Neural Nets WIRN Vietri-01*. 2002, Springer. p. 193-200.
11. Tian, Y. and L. Pan. *Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network*. in *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*. 2015. IEEE.
12. Cao, J. and X. Lin, *Study of hourly and daily solar irradiation forecast using diagonal recurrent wavelet neural networks.* Energy Conversion and Management, 2008. **49**(6): p. 1396-1406.
13. Kasabov, N.K. and Q. Song, *DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction.* IEEE transactions on Fuzzy Systems, 2002. **10**(2): p. 144-154.
14. Stach, W., L.A. Kurgan, and W. Pedrycz, *Numerical and linguistic prediction of time series with the use of fuzzy cognitive maps.* IEEE Transactions on Fuzzy Systems, 2008. **16**(1): p. 61-72.
15. Mocanu, E., et al., *Deep learning for estimating building energy consumption.* Sustainable Energy, Grids and Networks, 2016. **6**: p. 91-99.
16. Petrosian, A., et al., *Recurrent neural network based prediction of epileptic seizures in intra-and extracranial EEG.* Neurocomputing, 2000. **30**(1): p. 201-218.
17. Abdel-Hamid, O., L. Deng, and D. Yu. *Exploring convolutional neural network structures and optimization techniques for speech recognition*. in *Interspeech*. 2013.
18. Abdel-Hamid, O., et al., *Convolutional neural networks for speech recognition.* IEEE/ACM Transactions on audio, speech, and language processing, 2014. **22**(10): p. 1533-1545.
19. Kahou, S.E., et al. *Combining modality specific deep neural networks for emotion recognition in video*. in *Proceedings of the 15th ACM on International conference on multimodal interaction*. 2013. ACM.
20. Bengio, Y., A. Courville, and P. Vincent, *Representation learning: A review and new perspectives.* IEEE transactions on pattern analysis and machine intelligence, 2013. **35**(8): p. 1798-1828.
21. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.
22. Hubel, D.H. and T.N. Wiesel, *Receptive fields and functional architecture of monkey striate cortex.* The Journal of physiology, 1968. **195**(1): p. 215-243.
23. LeCun, Y., et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.
24. Waibel, A., et al., *Phoneme recognition using time-delay neural networks.* IEEE transactions on acoustics, speech, and signal processing, 1989. **37**(3): p. 328-339.
25. Lawrence, S., et al., *Face recognition: A convolutional neural-network approach.* IEEE transactions on neural networks, 1997. **8**(1): p. 98-113.
26. Taigman, Y., et al. *Deepface: Closing the gap to human-level performance in face verification*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.
27. Vaillant, R., C. Monrocq, and Y. Le Cun, *Original approach for the localisation of objects in images.* IEE Proceedings-Vision, Image and Signal Processing, 1994. **141**(4): p. 245-250.

28. Ciresan, D., et al. *Deep neural networks segment neuronal membranes in electron microscopy images*. in *Advances in neural information processing systems*. 2012.
29. Collobert, R., et al., *Natural language processing (almost) from scratch.* Journal of Machine Learning Research, 2011. **12**(Aug): p. 2493-2537.
30. Sainath, T.N., et al. *Deep convolutional neural networks for LVCSR*. in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 2013. IEEE.
31. Farabet, C., et al., *Scene parsing with multiscale feature learning, purity trees, and optimal covers.* arXiv preprint arXiv:1202.2160, 2012.
32. Hadsell, R., et al., *Learning long-range vision for autonomous off-road driving.* Journal of Field Robotics, 2009. **26**(2): p. 120-144.
33. Norouzi, M., et al. *Reward augmented maximum likelihood for neural structured prediction*. in *Advances In Neural Information Processing Systems*. 2016.
34. Mnih, V., et al. *Asynchronous methods for deep reinforcement learning*. in *International Conference on Machine Learning*. 2016.
35. Ioffe, S. and C. Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. in *International Conference on Machine Learning*. 2015.
36. Abadi, M., et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems.* arXiv preprint arXiv:1603.04467, 2016.
37. Mackey, M.C. and L. Glass, *Oscillation and chaos in physiological control systems.* Science, 1977. **197**(4300): p. 287-289.