Fourier-based Parametrization of Convolutional Neural Networks for Robust Time Series Forecasting

Sascha Krstanovic and Heiko Paulheim

Data and Web Science Group, University of Mannheim, Germany
{sascha, heiko}@informatik.uni-mannheim.de

Abstract. Classical statistical models for time series forecasting most often make a number of assumptions about the data at hand, therewith, requiring intensive manual preprocessing steps prior to modeling. As a consequence, it is very challenging to come up with a more generic forecasting framework. Extensive hyperparameter optimization and ensemble architectures are common strategies to tackle this problem, however, this comes at the cost of high computational complexity. Instead of optimizing hyperparameters by training multiple models, we propose a method to estimate optimal hyperparameters directly from the characteristics of the time series at hand. To that end, we use Convolutional Neural Networks (CNNs) for time series forecasting and determine a part of the network layout based on the time series' Fourier coefficients. Our approach significantly reduces the amount of required model configuration time and shows competitive performance on time series data across various domains. A comparison to popular, state of the art forecasting algorithms reveals further improvements in runtime and practicability.

Keywords: Time Series Forecasting · Neural Networks · Fourier Analysis

1 Introduction

In the age of connected sensors, devices, and services, temporal data is one of the most widespread data types these days. Designing accurate forecasting models typically involves lots of manual work, e.g. data preprocessing, parameter tuning, and model selection. Since time series data comes in different shapes and distributions, these manual steps are usually required for each new dataset.

In this work, we propose a time series forecasting framework based on CNNs that makes no prior assumptions about data distribution and integrates all required preprocessing steps. We demonstrate its predictive power on thirty data series, where the approach outperforms all baselines in two-thirds of the cases without the need to manually adapt any parameter across the different datasets. In addition to this, we show significant improvements in runtime of the training process, therewith, providing a very convenient forecasting method that is fast and robust.

Our approach combines the predictive power of CNNs with the time series decomposition capabilities of Fourier analysis. Hyperparameter tuning is expensive, because it requires training multiple models. In this paper, we follow a different approach and configure a neural network analytically. Our idea is based on the assumption that the characteristics of a time series – more specifically: its Fourier decomposition – can be used to determine a suitable network layout for a CNN. Hence, we exploit the inherent structure of time series data in order to parametrize the CNN used for forecasting.

Section 2 starts with existing approaches to time series forecasting, discussing their assumptions and strengths. In Section 3, we provide a detailed explanation of our contribution and its motivation. These ideas are applied to numerous real world datasets in Section 4, demonstrating advantages and limitations. We conclude and discuss future work in Section 5.

2 Related Work and Time Series Fundamentals

Due to the diverse occurrence of time series data in applications and databases, its analysis has been an active research field for decades. Temporal data has the interesting property that the current value is dependent on a number of past values. In other words, observations are not independent of each other but can be thought of as a function of their past values.

2.1 Autoregression and Smoothing

Autoregressive (AR) models are amongst the most popular approaches for time series analysis and forecasting. AR models approximate a time series with a linear combination of the most recent past values and their errors [4]. These models perform particularly well if the assumption is met that the series is generated by a linear process [1], however, this barely holds in practice.

Exponential Smoothing constitutes another relatively simple yet popular approach to forecasting. Here, the series is smoothed by applying an exponential window function. This implies the assignment of weights which decrease over time.

2.2 Machine Learning

Forecasting as a Supervised Regression Problem. An advantageous property of historical time series data is that transforming it to a supervised machine learning task is easy. Past observations serve as explanatory features to the respective future values that constitute the target variables. Unlike other supervised learning tasks (e.g. image recognition), time series data can be automatically transformed to a supervised problem without the need for manual annotation. This aspect is critical for the success of end-to-end forecasting frameworks such as the one presented in this paper.

Handled this way, the forecasting task follows the same process as any other supervised machine learning challenge, i.e., hyperparameter optimization, evaluation, and model selection. This idea is also implemented in [23] in order to train the base models required for ensemble learning.

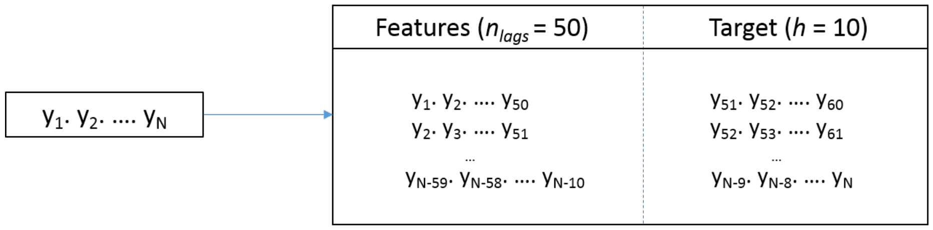
$y_1, y_2, \dots, y_N$
Features ($n_{lags} = 50$)
Target ($h = 10$)
$y_1, y_2, \dots, y_{50}$
$y_2, y_3, \dots, y_{51}$
...
$y_{N-59}, y_{N-58}, \dots, y_{N-10}$
$y_{51}, y_{52}, \dots, y_{60}$
$y_{52}, y_{53}, \dots, y_{61}$
...
$y_{N-9}, y_{N-8}, \dots, y_N$

Fig. 1. Transforming raw, univariate time series data to a supervised learning task

There are two common strategies of how to design a machine learning system for multi-step time series forecasting, known as the direct and indirect methods. Assuming a forecasting horizon $h > 1$, one can either train a dedicated model for each future point $1, 2, ..., h$ (direct method), or only train a single model and use its forecast as input for the succeeding future point in an iterated fashion (indirect method). Since the direct approach requires the training of h individual models it scales very poorly for longer horizons, leading to a low practicability for actual applications. It was also shown that the performance is inferior to that of the indirect method for AR models [9].

Artificial Neural Networks. [11] showed that an ANN with one hidden layer is able to approximate a continuous function arbitrarily well, which makes ANNs highly interesting for regression problems. The spike in popularity of ANNs within the past decade led to significant developments for time series analysis, especially with regard to recurrent neural networks (RNNs). As these models usually make no prior assumptions about data distribution, they have a major advantage over more classical time series models described in the previous section. Intuitively speaking, RNNs can be thought of as standard feed forward networks with loops in them. This sequential architecture enabled RNNs to achieve new state of the art results on a variety of sequential tasks such as machine translation [5] [20] and time series forecasting [3] [14] [17]. Nevertheless, RNNs tend to suffer from vanishing or exploding gradients in case of very long-term dependencies in the data [2] [18]. Long short-term memory (LSTM) cells overcome this problem by the introduction of a gating mechanism that regulates the information flow of the network [10]. This allows for more reliable modeling of long-term dependencies, leading to a wide adoption of LSTMs for sequential tasks.

Due to the sequential nature of RNNs, parallelization possibilities are limited and training these models is computationally expensive. As a consequence, the application of convolutional neural networks (CNNs) to sequential problems has been a very active field of study. CNNs rely on filtering mechanisms in order to generate meaningful meta-features out of the raw input data. While primarily used to analyze visual imagery [27] and audio [26], the application of CNNs to sequential regression problems has shown competitive prediction performance at significantly lower computational complexity [7].

2.3 Ensembles

An essential requirement for ensemble methods to be effective is that the base learners are heterogeneous and make different errors at prediction time. [22] presents strategies on diversity generation for ensemble models. [23] combines several heterogeneous machine learning base models that are arbitrated by a meta-learner to generate the final forecasts. [12] [13] make use of local minima in the LSTM training process by storing model snapshots every time the LSTM converges to a local minimum. While ensemble models boost predictive performance, training of multiple base learners leads to high computational costs.

3 Forecasting with CNNs

The general procedure to train a CNN forecasting model follows the steps required for any regression problem, where the lags of the input series serve as features and the respective future values as targets. Fig. 1 depicts the splitting logic in the data preprocessing stage. The forecasting horizon h is usually a given parameter that is defined by the problem at hand. More importantly, the number of past lags to include as features must be large enough in order to account for long-term relationships across the series.

3.1 A CNN Algorithm for Multi-Step Forecasting

While CNNs are best known for their powerful capabilities within the area of image recognition, they are also widely used for more traditional regression problems. Hence, their structure allows the application to autoregressive tasks such as time series forecasting. Contrary to RNNs, however, CNNs are not naturally built for sequence processing. In order to generate sequence forecasts despite its architecture, we apply the indirect forecasting method, following the logic described before.

3.2 Enhancing Parameter Optimization with Fourier Analysis

The major challenge when dealing with ANNs is hyperparameter optimization. As this is a data specific task, a robust forecasting framework greatly benefits from efficient parameter optimization.

For the special case of time series data, we make use of its inherent structure in order to enhance CNN parametrization. Since one of the key tasks of time series models is to correctly identify repeating patterns over time, it is essential to parametrize models in a way that enables them to determine these structures.

Fourier analysis is a powerful and efficient tool that solves the problem of determining autocorrelations [19]. We apply the fast Fourier transform (FFT) [21] to the training data and use the decomposition of the series to extract the strongest autocorrelations. As these frequencies determine the respective sizes of the strongest patterns, we can configure the convolutional layers so that they match those pattern lengths. Therefore, the CNN is directly tailored towards the patterns that dominate the time series under study. More precisely, the proposed method follows these steps to get from data input to forecasting:

1. z-standardize and transform the input data according to Fig. 1
2. Determine Fourier coefficients and top-2 autocorrelations
3. Define a 2-layer CNN and set the lengths of the convolution windows according to the top periodicities inferred from the Fourier coefficients. The CNN uses 96 filters, 50 past lags, a batch size of 16, a dropout rate of 20%, a learning rate of 10^{-4}, the Adam [8] optimizer, 100 epochs and mean squared error as loss function.
4. Train CNN and evaluate performance on the test set

4 Experimental Analysis

4.1 Baseline Models and Evaluation

In order to validate the performance of the proposed algorithm, we perform forecasting experiments on 30 real world datasets.

Methods. The 10-steps ahead forecasting accuracy, measured in terms of root mean squared error (RMSE), is compared to the following baseline methods (cf. Chapter 2 for details):

- ARIMA, where model selection is based on a parameter grid opting for the AIC ($p = 1, 2, 3$, $q = 1, 2, 3$, $d = 1, 2$)
- Exponential Smoothing (ES)
- Arbitrated ensembles (ArbEns) specified in [23]
- Fourier: continuing reconstructed Fourier signals to generate forecasts
- CNNs parametrized with common filter length selections $\{2, 8\}$ opposed to Fourier based parameter estimation (CNN-Std.)
- Standard 2-layer LSTM architecture as described in [13] (LSTM-Std.)
- LSTM with the same architecture as the previous one, trained using the Snapshot Ensemble approach from [13] (LSTM-Snap)
- Our proposed method, combining CNNs and Fourier analysis (CNN-Fou)

An implementation, written mostly in Python, is available on GitHub[1] . For arbitrated ensembles, the *tsensembler* library written in R is used since the authors released it in that language.

[1] https://github.com/saschakrs/CNN-Fou, accessed April 6, 2019

Data. The datasets listed in Table 1 are used for model training and validation. These time series originate from various domains such as air quality measurements and energy loadings. We also report the frequencies, mean values and standard deviations for each series. Furthermore, the length of each series is normalized such that all datasets have between 2.974 and 2.995 observations, making the results comparable.

In this analysis, we focus on the univariate case, i.e., forecasting the target variable based on its own past values. Prior to modeling, all data is standardized according to a z-transformation, i.e., $Y = \frac{Y-\mu}{\sigma}$. Note that mean μ and standard deviation σ are determined based on only the training set as the holdout data points are unknown in a real scenario. For each approach and dataset, the most recent 10% of the series are used as test data in a windowing fashion. Every model is evaluated based on its ability to provide accurate 10-step ahead forecasts.

Evaluation. The results are summarized in Table 2. For each method and dataset we report the forecast RMSE for the 10% holdout data sample.

In addition, we provide the runtimes for model training in Fig. 4.2. The neural networks were trained on a NVIDIA Tesla K80 GPU and an Intel i7-6820HQ CPU was used for all other models as these don't profit from GPU usage.

The key value of our method lies in its robustness across different datasets without the need to manually incorporate domain knowledge. This implies that we leave all training architectures and parameters constant for each series (except for the Fourier coefficients that are learned for each dataset). Therefore, the proposed method constitutes a framework for automated end-to-end time series forecasting that does not require additional, manual processing steps prior to modeling and forecasting.

4.2 Results

The results in Table 2 show that the proposed approach yields superior performance in 20 of 30 cases. We apply the Diebold-Mariano test [15] in order to evaluate whether the top performing model has a significantly different forecasting accuracy than the next best method. The null hypothesis states that the forecasting accuracy of the two methods are not different. One star (*) or two stars (**) indicate a p-value of less than 0.05 or 0.01, respectively. We can observe that:

- In two-thirds of all cases, the Fourier-integrated CNN is superior or equal to all baseline methods with an average performance gain of 10.25% compared to the next best method (Snapshot Ensembles).
- Traditional models such as ARIMA, exponential smoothing and simple Fourier forecasting show poor performance compared to advanced methods.
- In terms of runtime, modern CNN implementations benefit heavily from strong parallelization on GPUs. Here, LSTMs suffer from their sequential nature that makes them harder to train efficiently. Compared to the training of LSTMs, CNNs are faster by a factor of 4.

Table 1. Overview of Datasets

i	Domain	Frequency	μ	σ
1	Water consumption indicators in Porto [23]	30 min	1.75	1.38
2	Water consumption indicators in Porto	30 min	1079.03	271.90
3	Water consumption indicators in Porto	30 min	10.08	7.03
4	Solar radiation [23]	hour	930.72	289.03
5	Solar radiation	hour	112.46	86.14
6	Solar radiation	hour	912.74	279.47
7	Solar radiation	hour	84.25	45.84
8	Various air quality measurements [25]	hour	1593.23	366.28
9	Various air quality measurements	hour	948.45	380.95
10	Various air quality measurements	hour	20.29	7.89
11	Various air quality measurements	hour	43.41	17.38
12	Various air quality measurements	hour	1.00	0.30
13	Various air quality measurements	hour	15.96	44.31
14	Various air quality measurements	hour	27.82	35.80
15	Various air quality measurements	hour	202.37	287.44
16	Various air quality measurements	hour	84.96	122.25
17	Various air quality measurements	hour	72.77	105.03
18	Various air quality measurements	hour	58.06	29.89
19	Various air quality measurements	hour	40.35	16.90
20	Various air quality measurements	hour	33.42	18.81
21	Energy loads such as electricity or gas [23]	hour	1.67	0.74
22	Energy loads such as electricity or gas	hour	67.83	153.33
23	Energy loads such as electricity or gas	hour	2.15	0.46
24	Energy loads such as electricity or gas	hour	256.11	40.23
25	Energy loads such as electricity or gas	hour	1021.64	203.35
26	Exchange rates [24]	day	208.32	82.10
27	Rainfall in Melbourne [24]	day	522.93	87.65
28	Mean river flow [24]	day	456.74	105.85
29	Number of births in Quebec [24]	day	23.01	10.68
30	Mean wave height [24]	hour	4.49	3.16

– While arbitrated ensembles are amongst the top performers for each dataset, they are computationally expensive since a number of base learners must be trained in order for ensembles to be effective.
– Result significance differs depending on the dataset. This is due to varying problem complexity between datasets.

5 Conclusions

We presented an end-to-end time series forecasting framework based on CNNs and Fourier analysis which is more computationally efficient and accurate than existing approaches. We made use of the natural structure of time series data in order to capture repeating patterns effectively. It was shown that Fourier analysis can be used to enhance CNN parametrization and improve forecasting

Table 2. RMSE for 30 different datasets, 10-steps ahead forecasting

i	ARIMA	ES	ArbEns	Fourier	CNN-Std.	LSTM-Std.	LSTM-Snap	CNN-Fou
1	0.60	1.12	0.50	1.04	0.51	0.88	0.51	**0.38****
2	0.65	0.93	0.37	0.99	0.52	0.45	**0.21***	0.25
3	0.69	1.22	0.51	1.15	0.54	0.67	0.41	**0.39***
4	0.60	1.09	0.43	1.09	0.41	0.53	0.36	**0.31****
5	0.86	1.45	**0.53****	1.54	0.95	0.74	0.58	0.59
6	0.47	0.89	0.42	0.93	0.40	0.59	0.30	**0.30**
7	0.77	1.48	0.57	1.11	0.67	0.81	0.47	**0.45**
8	0.54	1.17	0.42	0.71	0.45	0.57	0.48	**0.30****
9	0.95	1.90	0.67	1.34	**0.40****	0.77	0.55	0.55
10	0.51	0.83	0.27	0.52	0.38	0.31	0.30	**0.27**
11	0.39	0.83	0.33	0.46	0.25	0.41	0.37	**0.21***
12	0.73	1.34	0.62	1.35	**0.48***	0.91	0.56	0.50
13	1.29	1.29	1.31	1.59	1.08	1.66	**1.01****	1.19
14	0.56	0.41	0.29	0.69	0.29	0.37	0.28	**0.27**
15	0.46	1.87	0.40	0.50	0.56	0.42	0.45	**0.40***
16	0.65	1.04	0.65	0.81	0.78	0.94	**0.46****	0.53
17	0.63	1.01	0.63	1.09	0.72	1.13	**0.50***	0.51
18	0.74	1.16	0.27	1.37	0.40	0.30	0.31	**0.26****
19	0.94	1.06	0.44	1.41	0.51	0.69	0.43	**0.32****
20	0.86	0.86	0.71	1.19	0.86	0.86	**0.42****	0.59
21	1.16	1.33	0.49	1.45	0.51	0.55	0.42	**0.37****
22	0.11	0.02	0.03	0.12	0.03	0.05	0.03	**0.03**
23	1.05	0.27	0.07	1.46	0.06	0.13	0.06	**0.05**
24	0.76	1.16	0.57	1.19	0.37	0.78	**0.44**	0.45
25	0.74	1.06	0.23	0.90	0.34	0.38	0.21	**0.18***
26	0.55	1.01	0.27	0.79	0.38	0.30	0.19	**0.15***
27	0.52	1.59	0.37	0.63	0.44	0.51	0.26	**0.25**
28	0.39	1.69	0.34	0.59	0.44	0.43	0.22	**0.22**
29	0.52	1.06	0.44	0.70	**0.23***	0.80	0.33	0.32
30	0.72	0.61	0.63	1.19	0.62	0.91	0.73	**0.55***
Avg	0.68	1.09	0.46	1.00	0.49	0.63	0.40	**0.37**

performance without the need to adapt the setup for new datasets. The method was compared to various state of the art forecasting methods and generated the most accurate results in the majority of thirty use cases.

While this work focused on the univariate case, its extension to the multivariate scenario will be part of our future research. The basic methodology will be the same, however, the additional amount of features requires more efficient preprocessing and modeling strategies. Apart from that, it is worth investigating the effects when scaling the framework to larger datasets, especially in terms of CNN architecture.

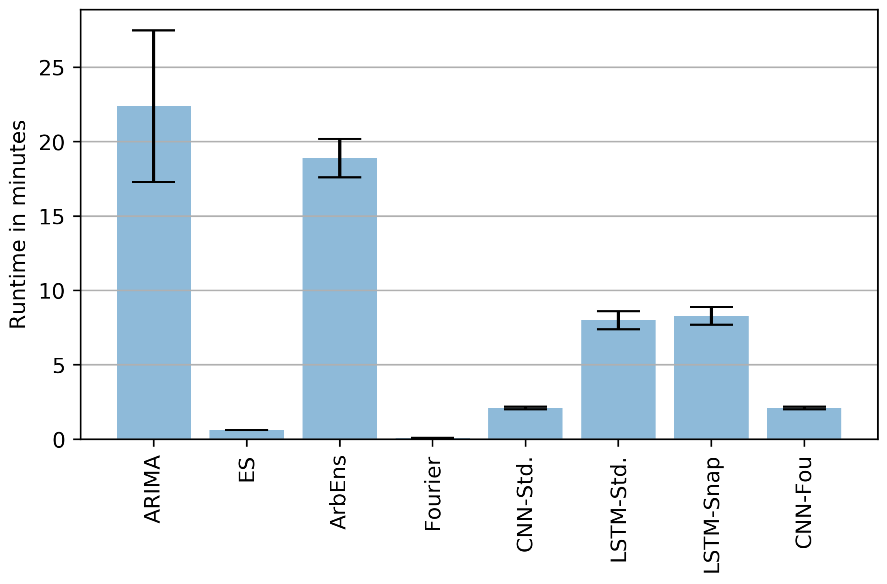

Fig. 2. Average model runtime across 30 datasets

References

1. Ratnadip Adhikari. 2015. A neural network based linear ensemble framework for time series forecasting. Neurocomputing 157 (2015), 231-–242.
2. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks 5, 2 (1994), 157—166.
3. Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. 2002. Applying LSTM to time series predictable through time-window approaches. In Neural Nets WIRN Vietri-01. Springer, 193—200.
4. James Douglas Hamilton. 1994. Time series analysis. Vol. 2. Princeton university press Princeton.
5. Zhen He, Shaobing Gao, Liang Xiao, Daxue Liu, Hangen He, and David Barber. 2017. Wider and Deeper, Cheaper and Faster: Tensorized LSTMs for Sequence Learning. In Advances in Neural Information Processing Systems. 1—11.
6. Keith W Hipel and A Ian McLeod. 1994. Time series modelling of water resources and environmental systems. Vol. 45. Elsevier.
7. Ashish Vaswani et al. 2017. Attention is all you need. In: Advances in neural information processing systems (pp. 5998-6008).
8. Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
9. Massimiliano Marcellino, James H. Stock, and Mark W. Watson. 2006."A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series. In: Journal of econometrics 135.1-2: 499-526.

10. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735—1780.
11. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. Neural networks 2, 5 (1989), 359—366.
12. Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Li, John Hopcroft, and Kilian Weinberger. [n. d.]. Snapshot Ensembles: Train 1 Get M for Free. In Proceedings of the International Conference on Learning Representations (ICLR 2017).
13. Sascha Krstanovic and Heiko Paulheim. 2018. Stacked LSTM Snapshot Ensembles for Time Series Forecasting. In Proceedings of ITISE 2018, International Conference on Time Series and Forecasting. Godel.
14. Martin Längkvist, Lars Karlsson, and Amy Loutfi. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters 42 (2014), 11-–24.
15. David Harvey, Stephen Leybourne, and Paul Newbold. 1997. Testing the equality of prediction mean squared errors. International Journal of forecasting 13.2: 281-291.
16. M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml
17. Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long short term memory networks for anomaly detection in time series. In Proceedings. Presses universitaires de Louvain, 89.
18. Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In International Conference on Machine Learning. 1310-–1318.
19. Deepak Sharma, Biju Issac, GPS Raghava, and R Ramaswamy. 2004. Spectral Repeat Finder (SRF): identification of repetitive sequences using Fourier transformation. Bioinformatics 20, 9 (2004), 1405-–1412.
20. Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems. 3104—3112.
21. Peter Welch. 1967. The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Transactions on audio and electroacoustics 15, 2 (1967), 70-–73. network model. Neurocomputing 50 (2003), 159-–175.
22. Mariana Oliveira and Luis Torgo. 2014. Ensembles for time series forecasting. JMLR: Workshop and Conference Proceedings 39:360—370
23. Vitor Cerqueira, et al. 2017. Arbitrated Ensemble for Time Series Forecasting. Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham.
24. Rob Hyndman: Time series data library.
https://datamarket.com/data/list/?q=provider:tsdl, accessed April 6, 2019
25. D. Dua and C. Graff. 2019. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, accessed April 6, 2019
26. Aäron Van Den Oord, et al. 2016. WaveNet: A generative model for raw audio. SSW.
27. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. nature 521.7553: 436.