Joint Fusion Learning of Multiple Time Series Prediction

Orçun Gümüş

***Abstract*—Accurate traffic density estimations is essential for numerous purposes like the developing successful transit policies or to forecast future traffic conditions for navigation. Current developments in the machine learning and computer systems bring the transportation industry numerous possibilities to improve their operations using data analyses on traffic flow sensor data . However, even state-of-art algorithms for time series forecasting perform well on some transportation problems, they still fail to solve some critical tasks. In particular, existing traffic flow forecasting methods that are not utilising causality relations between different data sources are still unsatisfying for many real-world applications . In this report, we have focused on a new method named joint fusion learning that uses underlying causality in time series. We test our method in a very detailed synthetic environment that we specially developed to imitate real-world traffic flow dataset. In the end, we use our joint-fusion learning on a historical traffic flow dataset for Thessaloniki, Greece which is published by Hellenic Institute of Transport (HIT) . We obtained better results on the short-term forecasts compared the widely-used benchmarks models that uses single time series to forecast the future.**

I. Introduction

ACCURATE traffic flow information is required to improve decision-making processes. By efficiently using traffic flow information individuals may decrease their daily transportation duration, public transports can improve their services by designing their scheduler more intelligently, or government reduce their carbon emissions and increase traffic operation efficiency [1], [2].

Accurate forecasting can also be very critical for individual basis. Suppose you are a tourist in Greece, and you want to travel from your Hotel to a famous Archaeological Museum of Thessaloniki by a car. You want to know the duration of this travel to plan the rest of your trip in the city. As you can see from the Figure 1, some possible trip routes between you and your target are highly crowded and request more time to cross then the usual duration. However this map shows a snapshot in the time domain, therefore after some period, states may vary, and the perfect path may be changed. So finding the best route possible not only requires the current knowledge but also future forecasts. Big companies like Google is also using traffic flow forecasts to not only calculate the shortest possible road trips from point A to B but also to estimate the paths with the shortest duration possible.

As we explained before traffic flows are not constant with time; they are generally more dense in prime hours of the day and more sparse in others. So estimating the shortest travel duration from point A to B is requiring not only to find a path

Advisors: Nguyen Thanh Tam, Karl Aberer

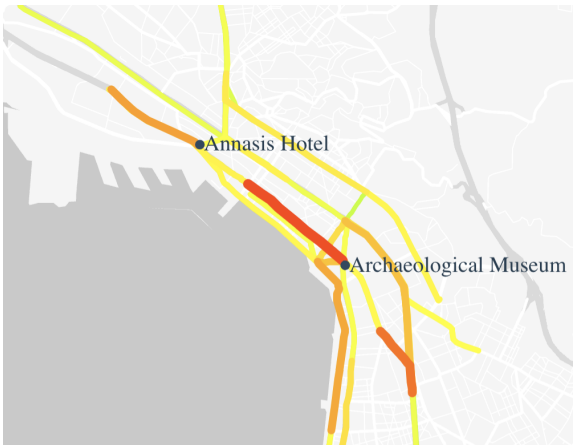

Figure 1: Prime time traffic flow map snapshot of Thessaloniki, Greece from 3th January 2018. Colors represent the normalized travel times. Red colors represent travel times that are closer to the monthly maximum values, greenish colors represent the travel times that are close the monthly minumum values. The travel times which are shown as gray represent missing values on the collected data.

in the graph but also forecasting future traffic flows of the road network. Many forecasting algorithms which are currently in use are only using a single time series to predict the future values of traffic flows [3]. In this work, we focus on using Joint-Fusion Neural Networks to obtain forecasts which can learn causality in the multiple time series.

We develop a very detailed synthetic environment to precisely control the variance and causality between different time series. We are inspired by real datasets while implementing the synthetic data generator by trying to include many different scenarios in the simulated environment. For testing our algorithm on a real traffic flow data, we use the open data provided from Hellenic Institute of Transport (H.I.T.) for the city of Thessaloniki, Greece. Using their API, we obtained the travel times of all the routes in Thessaloniki for every 15 min interval from January 2018 to February 2018 [4].

To develop a solution that uses causality in the different time series, we used generative LSTM networks with fusion-learning. Using this approach, we are able to predict future travel times more accurately than other baselines.

II. Problem Formulation

Definition II.1. (Forecasting, f) We donate the travel time of road segment k on time period t as x_k^t. Forecasting is the

prediction of future values using past values, finding $E(x_k^t)$ using $x_k^{t-i}, i < t$.

$$E(x_i^t) = f(x_i^{t-1}, x_i^{t-2}, x_i^{t-3}..., x_i^0) \quad (1)$$

We have used term joint fusion learning for forecasting multiple time series at the same time.

Definition II.2. (Joint Fusion Forecasting, g) We donate the travel times of all roads segment in the set on time period t as x^t.

$$x^t = [x_i^t, x_{i-1}^t, x_{i-2}^t..., x_0^t] \quad (2)$$

Joint Fusion Forecasting is the prediction of future values using past values of all timeseries, finding expected values of x^t , using $x^{t-i}, i < t$.

$$E(x^t) = g(x^{t-1}, x^{t-2}, x^{t-3}..., x^0) \quad (3)$$

III. Contribution

Time series forecasting becomes a harder problem when there is multiple time series to predict. We try to introduce a joint fusion learning method to capture the relations between different time series. Whereas other methods only focus on one-time series forecasting to the future of that time series, our method is tried to capture all intra-time series patterns.

A. Forecasting using Neural Nets

Artificial neural networks can learn high dimensional patterns via a sequence of non-linear transformations of input data; by allowing us to efficiently model of nonlinear functions.

Between different deep learning algorithms, recurrent neural networks, especially LSTMs, have been regularly used for forecasting purposes, and we also used LSTM neurons with our joint-fusion learning [5].

Let $X = [x^0, x^1, x^2..., x^t]$ timeseries with X^i vector of features at time point i. In this model there may be single or multiple feature in the timeseries vectors. Suppose there exists k different features and $X^i \in R^k$.

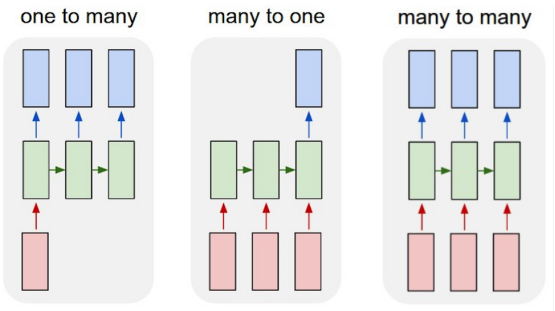

Figure 2: Different recurrent network structures [6]

There are multiple recurrent neural network structures for generative purposes. According to use cases, input and output shapes and dataset proprieties the most suitable structure of the neural network is varying. The three main different structural skeletons are demonstrated in Figure 2

1) one-to-many, single input with a sequence output (e.g. image captioning takes an image and outputs a sentence of words)
2) many-to-one, sequence input with a single output (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment)
3) many-to-many, sequence input with a sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French. [6])

We have implemented many-to-one and many-to-many approaches in order to compare the performance. In "many-to-one" scenario we have trained the model by giving all previous data in a window w and requesting the next value after the window. So our network outputs trained to fit $Y^{t+1} = x^{t+1}, X^{t+1} = [x^t, x^{t-1}, x^{t-w+1}...x^{t-w}]$.

In many to many scenario we used all previous outputs during training. So we used $Y^{t+1} = [x^{t+1}, x^t...x^{t-w+1}], X^{t+1} = [x^t, x^{t-1}, x^{t-w+1}...x^{t-w}]$.

During training of the neural network, we have used first-order gradient-based optimisation of stochastic objective functions, based on adaptive estimates of lower-order moments (ADAM), and we have used backpropagation to find optimal parameters for the model weights [7].

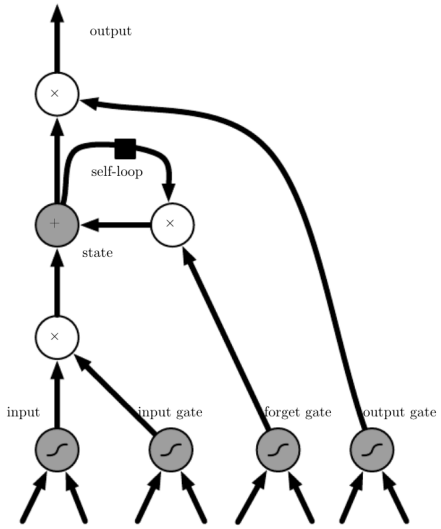

Figure 3: Block diagram of the LSTM recurrent network "cell." Cells are connected recurrently to each other, replacing the usual hidden units of ordinary recurrent networks. The state unit has a linear self-loop whose weight is controlled by the forget gate. The output of the cell can be shut of by the output gate. All the gating units have a sigmoid nonlinearity [8]

For long input sequences, RNN structure has some issues such as vanishing gradient. LSTM address such difficulties by designing a neuron model that can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing constant error flow through constant error carrousels [9]. LSTM consists of multiple functions that try to remember the helpful and forget the unnecessary information from inputs. An example of LSTM neuron is given in Figure 3.

B. Joint Fusion Learning

We supposed that traffic density on different streets in a particular city is correlated. So knowing a traffic density of a street is important information about another street. So we can assume that traffic densities on different roads are not independent. In other words $P(X_i^t, X_j^t) \neq P(X_i^t) \times P(X_j^t), i \neq j$.

Let us assume graph G is a graph as denoted as $G = (V, E)$, where E denotes a correlation between two time series and V indicate the travel time series for a street. We construct this graph using 0.5 for correlation threshold. Then we run the spin-glass community detection algorithm on graph G to find correlated travel times.

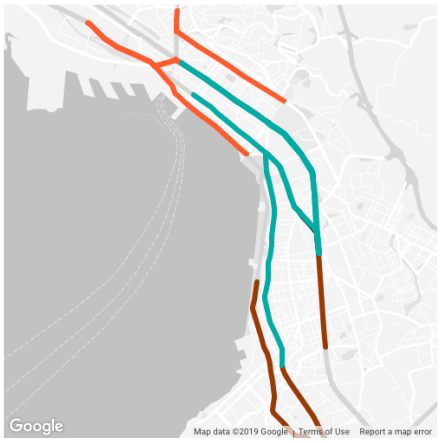

Figure 4: Detected road communities in Thessaloniki, Greece. Each different color represent a group. Roads in same communities varying together; in other worlds travel times are more correlated.

We are using multiple time series to improve prediction accuracy. For example, we can consider that time series that represent traffic density in a road may be heavily correlated with past of others. This means X_i^t may be correlated with X_j^{t-k}. In this formulation, X_j^{t-k} represent k step previous value of X_j^t. So in order to improve prediction accuracy, we may want to use these dependencies that are named as Granger causality. That means if past values of X_i^t helps in predicting future values of X_j^t we say X_i^t granger causes X_j^t [10].

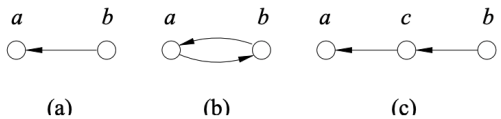

Figure 5: Causality patterns: (a) direct causality, (b) direct feedback, (c) indirect causality

In this section, we outline our proposed "Joint Fusion Learning", a scheme which is based on neural network. Our aim in this formulation is capturing Granger causality relationships in between different time series. Let us modify graph G, by changing edges to represent not the correlation for that time, but the correlation for one step lagged version ($X_i^t \sim X_j^{t-1}$).

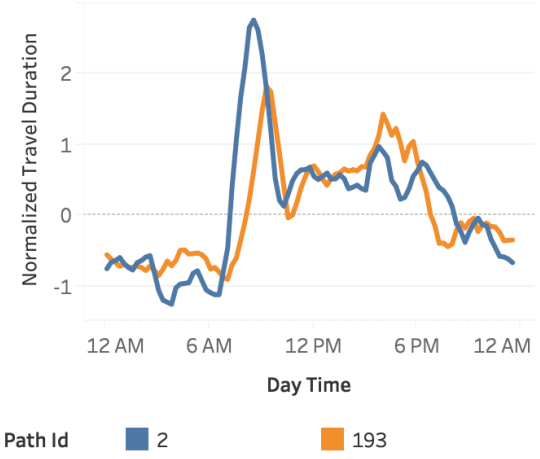

Figure 6: Normalized travel duration for street 193 (Egnatia) and street 2 (Konstantinou Karamanli) for 10th January 2018

For example in the Figure 6, at prime times of 10 January 2018 we first see a peak in street-2 then with a 15-minute lag street-193 also becomes crowded. So street-2 can help to predict better street-193. There may be many possibilities that may because that type of lagged causalities; for example, suppose there is a sporting event that finishes at 20:00 at a sports arena close to the street-2. When the event finalised, everybody is leaving the sports arena . So as a result of this we first see a local peak in the street-193 then a step further this may cause other peaks in other streets like street-2.

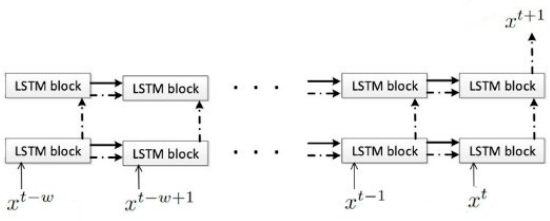

Figure 7: Two Layer Joint Fusion Neural network model; LSTM neuron type with many-to-one structure, $x^t = [x_0^t, x_1^t, ...x_k^t]$. Layers added on top of the other layers.

To learn these types of causalities and take advantage of lagged correlations we use joint fusion learning. So for example the model for forecasting the time $t+1$ model is taking all inputs from $t-w$ to t and a generate a point estimation for $t+1$. If we have k different roads in our model, the input for our algorithm is a k-dimensional vector at each time step, and we forecast a k-dimensional vector for the next time step.

IV. Experimental Results

A. Setup

1) Real Datasets: The traffic flow dataset has been collected from CERTH-HIT OpenData Hub BETA web site using their JSON API [4]. The data contains travel times of main streets of Thessaloniki, Greece for each 15 min periods from January to 2018 to February 2018. The data formatted in CSV and contains only the durations in the second format. To visualise roads on the map, we have used another dataset from the same portal which contains shapefile for the paths. The shapefile contains geometry of the streets by multiple points in lattitude and longitude format.

2) Synthetic Datasets: In order to test our forecasting method, we have designed a synthetic data generator. We develop it in such a way that it can imitate the real world scenarios. To do so, we have implement a very detailed random walk generator that can handle multiple correlated time series. We become able to generate various datasets from a basic random walk to a mixture of many different patterns. Before going deep into the synthetic dataset results, we introduce you the patterns that you can use to generate random walks:

Gaussian Random Walk (GRW). The Gaussian random walk is the sum of a series of independent and identically distributed random variables; x_i taken from a normal distribution with mean equal zero and variance σ^2

$$x^i = \mathcal{N}(0, \sigma^2)$$
$$X^t = \sum_{i=1}^{t} x^i$$

Correlated Gaussian Random Walk (CGRW). We expect that our model should be able to learn the causality between different time series. To test it, we are using correlated time series with a constant lag. To explain it more precisely suppose we have two different Gaussian random walks and x_1^i and x_2^i are i th steps of first and second time series respectively. x_1 and x_2 are the vectors from one multivariate normal distribution for x, so $x = (x_1, x_2)^T$.

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \sim N \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & c \\ c & 1 \end{pmatrix} \right]$$

Then we are using this multivariate normal distribution to create two timeseries with a lagged corralation to represent causality. So if we suppose lag constant as l the new timeseries X_1^t and X_2^t will be:

$$X_1^t = \sum_{i=1}^{t} x_1^i \tag{4}$$

$$X_2^t = \begin{cases} \sum_{i=1}^{t-l} x_2^i, & \text{if } t > l. \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

As you can see from Equation 5 there is a lag constant l which makes possible predict better X_2^t using X_1^{t-l}.

Patterns as Short Random Walks. There are repetitive patterns in time series of Trrafic Flow Data [2]. So in order to imitate this structure, we have used short random walk to represent repetitive patterns along the time series. The short random walks can be think as similar version of wavelet in a wave.

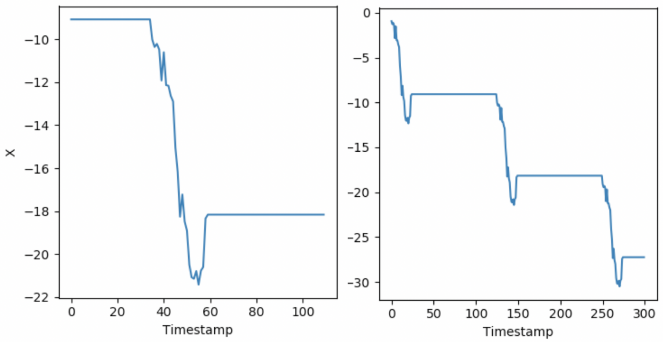

Figure 8: On the left side of the figure, there is an example of a short random walk, on the right side, we demonstrate how we use this short random walk on a time series. To illustration purposes, we use the constant inter-gaps between short walks, but in reality, the gaps in between short walks are uniformly distributed.

We have also come up with an idea of using patterns across different series to imitate causality between time series. For example, if a specific pattern occurs on X_1^t, we may assume that another specific pattern will occur on $X_2^(t+l)$. We have used this approach on top of correlated random walks to generate causality between series.

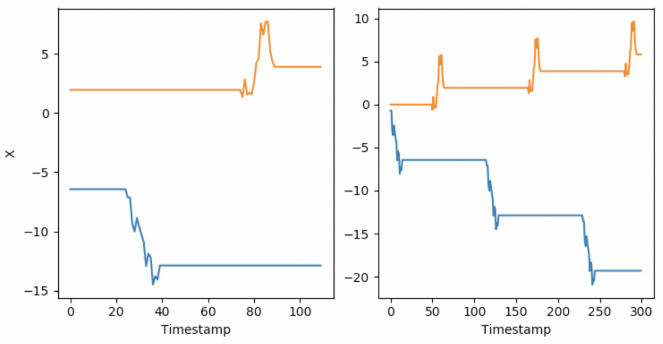

Figure 9: On the left side of the figure, there is an example of two short random walk with a intra-gap, on the right side, we demonstrate how we use this pair on two two time series.

Table I: Used definitions

Symbols	Definitions
MA	Moving Average Method
ES	Exponential Smoothing
DES	Double Exponential Smoothing
RNN	RNN network without Joint Fusion Learning
$J-RNN$	RNN network with Joint Fusion Learning
$LSTM$	LSTM network without Joint Fusion Learning
$J-LSTM$	LSTM network with Joint Fusion Learning
CRW	Correlated Random Walk
$Mixture$	Random Walk with trend and seasonal component
$Patterns$	Random walk with constant patterns

3) Metrics: There are many different metrics to measure the error in machine learning. We have used three measurement technique to demonstrate the comparison. The reason that we have selected NMSE over MSE is that we have two different dataset that have different scales for predictions. R^2 method is also a measurement which is not scale sensitive.

- *MAE:* Metric which measures the average magnitude of the errors in a set of predictions, without considering their direction.

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{6}$$

- R^2*:* The coefficient of determination, denoted R2 or r2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable.
- *Normalized MSE:* Normalized (Root) Mean Squared Error which facilitates the comparison between datasets or models with different scales.

$$\text{NMSE} = \frac{\sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2}}{max(Y)-min(Y)} \tag{7}$$

4) Baselines: We have used different forecasting methods as baselines to test our success and compare the results. One of the most simple technique in time series forecasting is taking the last value of the sequence as the future estimation. We call this as Persistence Method, as we naively assume the same last value for the future.

$$f_{persistence}(x_i^{t-1}, x_i^{t-2}, ..., x_i^0) = x_i^{t-1} \tag{8}$$

One of the most used forecasting technique in time series is taking average of the last n values as the future prediction. This method named as Moving Average in the literature [11].

$$f_{MA(n)}(x_i^{t-1}, x_i^{t-2}, ..., x_i^0) = \frac{1}{n}\times\sum_{k=1}^{n}x_i^{t-k} \tag{9}$$

Exponential smoothing is another technique that we use to forecast time series. Whereas in the moving average the past observations are weighted equally, exponential functions are used to assign exponentially decreasing weights over time. The constant a represent smoothing constant in the formula [11].

$$f_{ES(a)} = a(x_i^{t-1} + (1-a)x_i^{t-2} + (1-a)^2x_i^{t-3}...) + (1-a)^{t-1}x_i^0$$

Double exponential smoothing is a technique that we used to forecast time series with a trend. Whereas in the exponential smoothing only process level in the time series, double exponential smoothing also handles slope of the time series by exponentially smoothing it [11].

$$s^1 = x^1$$
$$b^1 = x^1 - x^0$$

$$s^t = ax^t + (1-a)s^{t-1} + b^{t-1}$$
$$b^t = bs^t - s^{t-1} + (1-b)b^{t-1}$$

$$f_{2ES(a,b)} = s^{t-1} + b^{t-1}$$

All these different forecasting methods have been implemented and used for performance evaluation purposes. Some ways perform better in some specific scenarios whereas some are valid in others. We have also compared different neural network models in varying structures. All these implementations will be discussed with positive and negative outcomes in the results section.

B. Experiments on Traffic Flow Data

The results on synthetic datasets shows that joint fusion neural networks are capable to learn causality relasionship between different timeseries. By using this result, we used the same model to forecast future values of traffic flow data. For the seek of simplicity we have focused on two roads which are highly corralated. To find this two roads we have shifted travel times with one time step and calculate the pairwise corralations then we have selected the pair with the highest correlation. The maximum pairwise correlation that we have calculated is 0.65.

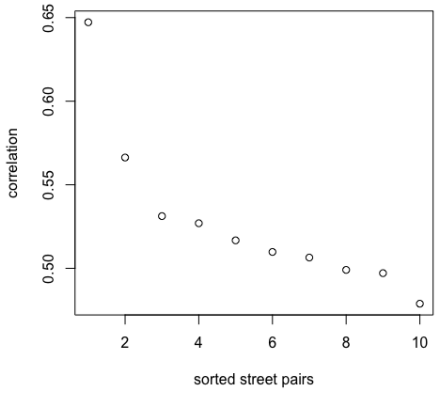

Figure 10: The travel times of road pairs sorted by their correlation. Visualized only ten road pairs which have maximum pairwise correlation.

The street pair which have maximum correlation (193 and 2) are used in our experiments. Joint Fusion Learning is capable

of learning causual relationships which fall into more then two series.

As explained earlier, by implemeted joint-fusion models we use information of two timeseries to forecast one, in order to improve the forecasting performance as compared to the case when we only use one time series. As you can see from the Table II, Joint-Fusion model have slightly better error measurements. Even the model J-LSTM uses exactly the same neural network structure with the LSTM model (same gates, same layer count, same neuron count, same activation function etc) it has achieved better results. The estimations and probability density function of the residuals of different methods are visualized and can be examined in Figure 11 and Figure 12.

Table II: Average Errors of different methods on traffic flow dataset of Thessaloniki, Greece

Methods	R^2	MAE	Normalized MSE
MA	0.6910	805.8124	0.0828
ES	0.7091	758.6089	0.0803
DES	0.6867	816.9235	0.0833
LSTM	0.7308	702.0266	0.0772
J-LSTM	**0.7403**	**678.5740**	**0.0759**

C. Experiments on Simulations

In this section, we discuss the results on the simulated datasets created using the different approaches demonstrated. In our experiments, we have used $l = 1$ to generate correlated random walks. The activation function of the neural network is set to 0.0001 and data is normalized between 0 and 1. We have used different neural network models(RNN, LSTM, MLP) that have one or more hidden layers. To compare the performance of our algorithm we use three different error measure MAE, R^2, MSE and Normalized MSE.

We have generated simulated datasets which contain correlated time series to test our algorithm. The dataset only contains 2 different time series with 0.5 correlation with one step offset on top of different trends and seasonal patterns.

Table III: Average Errors of different methods on synthetic dataset

Methods	R^2	MAE	Normalized MSE
MA	0.9986	1.7575	0.0122
ES	0.9983	2.2013	0.0136
DES	0.9988	1.5312	0.0114
LSTM	0.9989	1.3339	0.0106
J-LSTM	**0.9991**	**1.0884**	**0.0096**

The results of synthetic datasets have been shared in Table III. Among all methods, our implemented method, J-LSTM perform best compared with others. All parameters for forecasting methods have been determined using grid search for best performance on normalised mean square error. Exponential smoothing applied with $a = 0.8$, double exponential smoothing applied with $a = 0.8, b = 0.5$, moving average method has been applied using using $w = 1$.

As it has been demonstrated before, the simulated dataset model also has a sinusoidal and polynomial signal. This sinusoidal and polynomial signal makes it perfect to be learned by neural networks. As a result of this, neural network models such as LSTM perform better than usual forecasting methods like ES or DES. However even standard neural networks models can learn trend, and sinusoidal signal they are not able to catch the correlation between different time series. Our joint fusion learning is also catching correlation information, and error becomes diminutive compared to other forecasting algorithms.

V. Related Work

Similar ideas of joint-fusion have been found in various domains such as data integration [12], [13], [14], [15], [16], human computation [17], [17], [18], [19], [20], [21], streaming query processing [22], recommendation systems [23], web credibility [24], data exploration [25], information retrieval [26], sensor data [27], and financial time series [28].

VI. Conclusions and Future Work

We introduce a joint-fusion learning method using long-short-term memory neural networks. In our suggested method, we are able to catch the Granger causality information of correlated time series. Mainly, we focus on improving forecasting capabilities of deep learning methods in time series. To do so instead than using single time series historical data we fed the neural network with different correlated time series.

After developing joint-fusion learning method, we implement very detailedly and precise synthetic time series generator to test the forecasting abilities of different methods. We are able to generate time series which contains consistent patterns on correlated walks or sinusoidal seasonality or polynomial trends or even a mixture of all of them. Using controlled experiments on the generated dataset, we become sure that joint-fusion learning method can learn the Granger causality between different time series .

We use joint-fusion learning method on traffic flow datasets to forecast future travel times of different streets. Joint-fusion model improve the forecasting accuracy over the standard neural network models. In our testings, we only focus two-time series which have a causality relation, to increase the success of the joint-fusion forecasting more time series can be used at the same time. The dataset that we are using only contains major roads and streets. As we are using neural networks other data sources can be used in this system as additional features, for example a more detailed dataset that contains more roads and information like road density may be better for forecasting. The joint fusion method that we have discussed in this report can also be used in other data domains that have causality relation in it.

References

[1] C. P Ij Van Hinsbergen, J. Lint, and F. M Sanders, "Short term traffic prediction models," *14th World Congress on Intelligent Transport Systems, ITS 2007*, vol. 7, 11 2007.

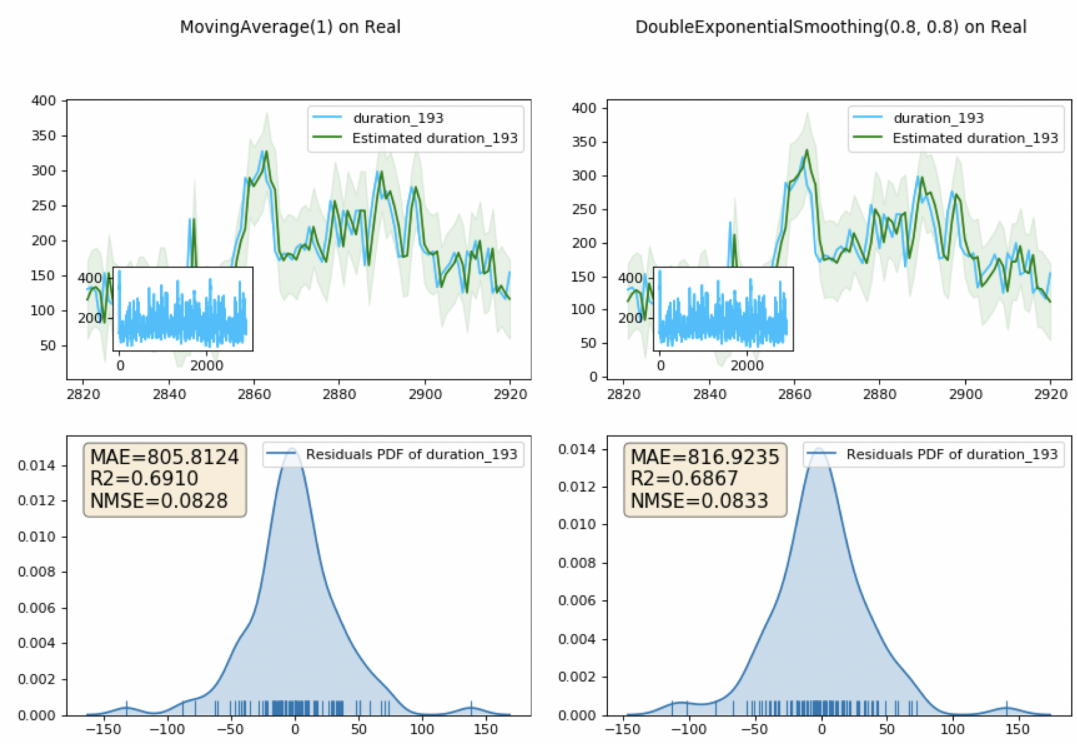

Figure 11: Results of Moving Average, Double Exponencial Smothing. Green lines represent estimations blue lines are real vales. For visualization purposes graphs are focused on last 100 steps. We also visualize the probability density functions of residuals using kernel as Gaussian distribution.

[2] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, April 2015.

[3] B. Said, J. Benhra, and H. El Hassani, "Causal method and time series forecasting model based on artificial neural network," *International Journal of Computer Applications*, vol. 75, pp. 37–42, 08 2013.

[4] H. Hellenic Institute of Transport, "Traffic flow," 2018, data retrieved from H.I.T. Portal, http://opendata.imet.gr/dataset.

[5] A. Ghaderi, B. M. Sanandaji, and F. Ghaderi, "Deep forecast: Deep learning-based spatio-temporal forecasting," *CoRR*, vol. abs/1707.08110, 2017. [Online]. Available: http://arxiv.org/abs/1707.08110

[6] "The unreasonable effectiveness of recurrent neural networks," http://karpathy.github.io/2015/05/21/rnn-effectiveness/, accessed: 2019-01-2.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[8] "Long-short term memory and other gated rnns," https://cedar.buffalo.edu/~srihari/CSE676/10.10%20LSTM.pdf, accessed: 2019-01-2.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[10] R. Dahlhaus and M. Eichler, "Causality and graphical models in time series analysis," *Oxford Stat. Sci. Ser*, vol. 27, 01 2003.

[11] J. Armstrong, "Selecting forecasting methods," *SSRN Electronic Journal*, 12 2009.

[12] N. Q. V. Hung, N. T. Tam, Z. Miklós, K. Aberer, A. Gal, and M. Weidlich, "Pay-as-you-go reconciliation in schema matching networks," in *ICDE*, 2014, pp. 220–231.

[13] N. T. Tam, N. Q. V. Hung, M. Weidlich, and K. Aberer, "Result selection and summarization for web table search," in *ICDE*, 2015, pp. 231–242.

[14] N. Q. V. Hung, X. H. Luong, Z. Miklós, T. T. Quan, and K. Aberer, "Collaborative schema matching reconciliation," in *CoopIS*, 2013, pp. 222–240.

[15] N. Q. V. Hung, T. K. Wijaya, Z. Miklos, K. Aberer, E. Levy, V. Shafran, A. Gal, and M. Weidlich, "Minimizing human effort in reconciling match networks," in *ER*, 2013, pp. 212–226.

[16] A. Gal, T. Sagi, M. Weidlich, E. Levy, V. Shafran, Z. Miklós, and N. Q. V. Hung, "Making sense of top-k matchings: A unified match graph for schema matching," in *IIWeb*, 2012, p. 6.

[17] N. Q. V. Hung, N. T. Tam, N. T. Lam, and K. Aberer, "BATC: a benchmark for aggregation techniques in crowdsourcing," in *SIGIR*, 2013, pp. 1079–1080.

[18] N. Q. V. Hung, C. T. Duong, N. T. Tam, M. Weidlich, K. Aberer, H. Yin, and X. Zhou, "Argument discovery via crowdsourcing," *VLDB J.*, pp. 511–535, 2017.

[19] N. Q. V. Hung, H. H. Viet, N. T. Tam, M. Weidlich, H. Yin, and X. Zhou, "Computing crowd consensus with partial agreement," *TKDE*, pp. 1–14, 2018.

[20] N. Q. V. Hung, D. C. Thang, N. T. Tam, M. Weidlich, K. Aberer, H. Yin, and X. Zhou, "Answer validation for generic crowdsourcing tasks with minimal efforts," *VLDB J.*, pp. 855–880, 2017.

[21] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer, "Minimizing efforts in validating crowd answers," in *SIGMOD*, 2015, pp. 999–1014.

[22] N. T. Tam, M. Weidlich, D. C. Thang, H. Yin, and N. Q. V. Hung, "Retaining data from streams of social platforms with minimal regret," in *IJCAI*, 2017, pp. 2850–2856.

[23] H. Yin, L. Chen, W. Wang, X. Du, N. Q. V. Hung, and X. Zhou, "Mobisage: A sparse additive generative model for mobile app recommendation," in *ICDE*, 2017, pp. 75–78.

[24] T. T. Nguyen, T. C. Phan, Q. V. H. Nguyen, K. Aberer, and B. Stantic, "Maximal fusion of facts on the web with credibility guarantee," *Information Fusion*, vol. 48, pp. 55–66, 2019.

[25] N. Q. V. Hung, K. Zheng, M. Weidlich, B. Zheng, H. Yin, N. T. Tam, and B. Stantic, "What-if analysis with conflicting goals: Recommending data ranges for exploration," in *ICDE*, 2018, pp. 1–12.

[26] N. T. Toan, P. T. Cong, N. T. Tam, N. Q. V. Hung, and B. Stantic, "Diversifying group recommendation," *IEEE Access*, vol. 6, pp. 17 776–17 786, 2018.

[27] N. Q. V. Hung, H. Jeung, and K. Aberer, "An evaluation of model-based approaches to sensor data compression," *TKDE*, pp. 2434–2447, 2013.

[28] N. Q. V. Hung and D. T. Anh, "Combining sax and piecewise linear approximation to improve similarity search on financial time series," in *ISITC*, 2007, pp. 58–62.

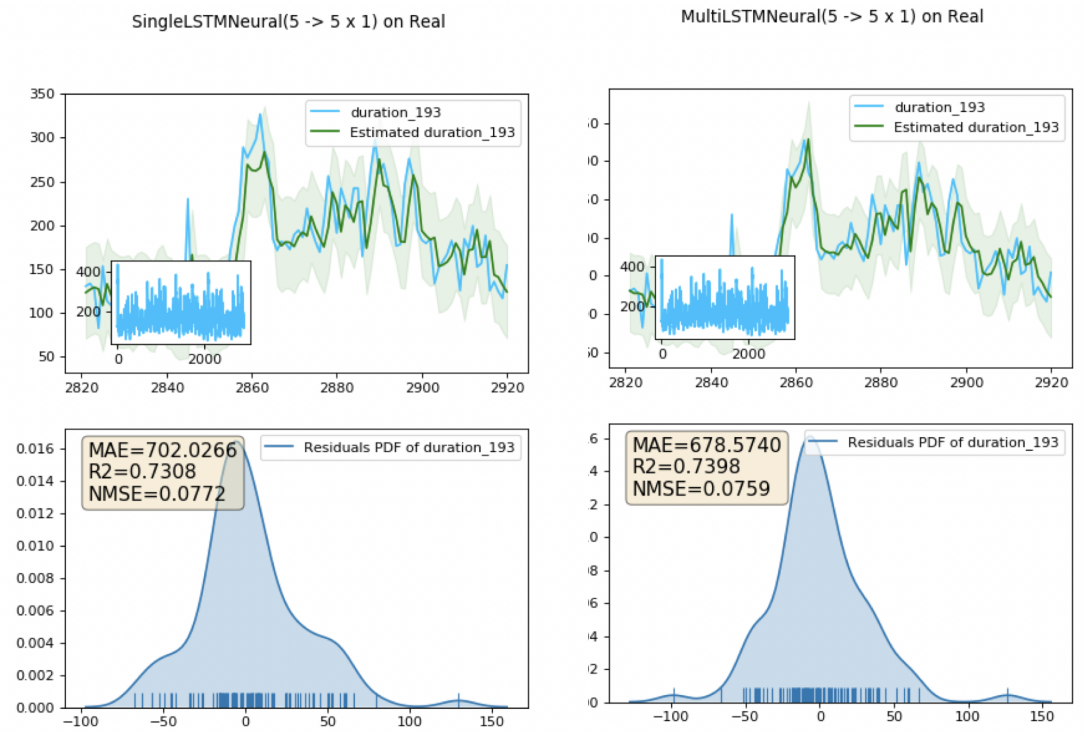

Figure 12: Results of LSTM neural network and Fusion LSTM neural network. Green lines represent estimations blue lines are real vales. For visualization purposes graphs are focused on last 100 steps. We also visualize the probability density functions of residuals using kernel as Gaussian distribution.