Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 184 (2021) 283–290

Procedia Computer Science

www.elsevier.com/locate/procedia

The 12th International Conference on Ambient Systems, Networks and Technologies (ANT)
March 23 - 26, 2021, Warsaw, Poland

Forecasting Public Transport Ridership: Management of Information Systems using CNN and LSTM Architectures

Sergey Khalil[a,b,*], Chintan Amrit[b], Thomas Koch[a,c], Elenna Dugundji[a,c]

[a] *CWI - National Research Institute for Mathematics and Computer Science, Amsterdam Science Park 123, 1098 XG Amsterdam, Netherlands*
[b] *Universiteit van Amsterdam, Amsterdam Business School, Plantage Muidergracht 12, 1018 TV Amsterdam, Netherlands*
[c] *Vrije Universiteit Amsterdam, Faculty of Science, De Boelelaan 1111, 1081 HV Amsterdam, Netherlands*

Abstract

This research paper provides a framework for the efficient representation and analysis of both spatial and temporal dimensions of panel data. This is achieved by representing the data as spatio-temporal image-matrix, and applied to a case study on forecasting public transport ridership. The relative performance of a subset of machine learning techniques is examined, focusing on Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) neural networks. Furthermore Sequential CNN-LSTM, Parallel CNN-LSTM, Augmented Sequential CNN-LSTM are explored. All models are benchmarked against a Fixed Effects Ordinary Least Squares regression. Historical ridership data has been provided in the framework of a project focusing on the impact that the opening of a new metro line had on ridership. Results show that the forecasts produced by the Sequential CNN-LSTM model performed best and suggest that the proposed framework could be utilised in applications requiring accurate modelling of demand for public transport. The described augmentation process of Sequential CNN-LSTM could be used to introduce exogenous variables into the model, potentially making the model more explainable and robust in real-life settings.

Keywords: Ridership Forecasting; Public Transportation; Machine Learning; CNN; LSTM; Neural Nets

1. Introduction

1.1. Problem statement and background

In the context of public transport, forecasting ridership is essential for business operations. Knowing the demand for travel allows public transport companies to make the transportation system more efficient, and helps to proactively improve the level of service for their customers and eliminate unnecessary costs [13, 1]. Having an effective model of how demand for public transport changes over time is important for future planning, introducing new routes, more efficient schedules, and optimising transportation operations (e.g. frequency of buses on a certain line) [3, 5].

* Corresponding author. Email: srg.khalil@gmail.com

1877-0509
10.1016/j.procs.2021.03.037

This research aims to improve demand modeling for public transport based on historical ridership data. The main objective of the research is to showcase different combinations of machine learning models in order to effectively account for both spatial and temporal relationships in historical ridership data. These models are trained and then tested on how accurate the forecasts are.

The proposed methodology is a combination of two prominent neural network techniques: Convolutional Neural Networks (CNN) are used in image processing (recognition and classification), and Long Short Term Memory (LSTM) models are great at identifying temporal patterns in data. More specifically, this research provides a performance comparison of 5 different neural network architectures: CNN only, LSTM only, Sequential CNN-LSTM, Parallel CNN-LSTM as well as Augmented Sequential CNN-LSTM architectures. Additionally, a classical statistical baseline approach is showcased in the form of a Fixed Effects Ordinary Least Squares (OLS) model for panel data.

It is not immediately apparent that there would be a substantial difference in performance depending on the way that the models are fit together. Yet, this research will demonstrate that there indeed is significant difference and will provide possible explanations of why this may be the case.

1.2. Scope of the research and research question

The research is conducted on a large spatio-temporal data set counting quarterly ridership across all public transportation operators in the greater Amsterdam region. The scope of the research is limited to a relatively small subset of available Origin-Destination (OD) pairs in the original data set. The main reason for it is the way that the data had been aggregated by the Vervoerregio [1] due to privacy concerns. Only the OD pairs that have relatively large ridership values (more than 300 rides per day of the week in each quarter in the whole period) have been considered.

Furthermore, this research has been conducted within the framework of the *Impact Study North-South Line*, which means that one of the goals has been to specifically look into how ridership had been affected by the inauguration of the North-South metro Line (NSL) in Amsterdam. The implication of this is that research focuses on routes to and from the North of the IJ.

Also, this research will not focus on exploring *all* possible architectural choices for the CNN, LSTM and combined CNN-LSTM models. Instead, a baseline architecture for all is chosen so that the relative performance of the models in question is highlighted. It is important to mention that this research does not assume these architectures to be the best machine learning algorithms, but rather such selection of algorithms and combined architectures is based on the review of relevant literature, builds and improves on previous research applying CNN, LSTM and combined architectures to forecasting ridership.

The proposed methodology mainly focuses on efficient input processing, however the described models could easily be adapted to a different forecasting goal, such as: predicting categorical ranges, binary flags, identifying specific cases in data. This would be just a matter of assigning proper labels to a given sample set.

The main research question is to determine whether a CNN could be combined with a LSTM model to effectively extract both spatial as well as temporal relationships from the available ridership data, and if that is the case, showcase which architecture captures these relationships best.

2. Review of relevant literature

2.1. Convolutional Neural Networks (CNN)

CNN have shown outstanding results in such tasks as image recognition, image classification, and computer vision [6, 12]. One of the advantages of CNN architecture is that convolution layers are sparsely connected to the input, instead of fully connected as in the case of simple neural networks. This allows the possibility for individual filters to extract specific features more efficiently.

Another significant feature of CNN is the introduction of pooling layers, which in the case of Max-Pooling selects the most prominent features in a given sample. These two characteristics (sparse connection to input and pooling

[1] Transportation authority of the greater Amsterdam region

layers) drastically reduce the number of modelled parameters and thus allows CNN to be employed on classification problems on a large scale [4]. At the same time research in the field of face recognition shows that these characteristics allow CNN to efficiently incorporate spatial relationships in the input data [7].

The CNN model can be described as an automatic feature extractor for different kinds of input data [8]. This flexibility of CNN to extract features from different sources is also shown in the field of music information retrieval by Li et al. [9], where CNNs are employed for capturing variations in musical patterns for music genre classification. In fact, authors suggest that there was almost no modification needed to adapt the CNN to this particular task.

In order to apply CNN for feature extraction for forecasting ridership, the model required some adjustment. Namely, the model inputs would be slightly different since values in the ”image” will be representing ridership values at a particular time on a particular route, instead of the level of color intensity for a given pixel of an image. Ridership values needed to be scaled to an interval from 0 to 1 to prevent difficulties in model training.

In the context of forecasting ridership, CNN outputs also need to be conceptualised differently - CNN will be predicting ridership instead of an image class (as is usually the case with image classification). The loss function also had to be adjusted in accordance with the goal of predicting continuous ridership values.

2.2. Long Short-Term Memory (LSTM) and combined Neural Networks

Long Short-Term Memory networks have been used in a variety of contexts and use cases. One of which is music, more specifically music computation as a dynamic information system described by Franklin [2]. LSTM neural networks are well suited for the task of learning patterns in music and reproduction of songs is that the state of a given song is conceptualised as a complex system that depends on past states. This strength of LSTM neural networks potentially would be useful in the task of modelling demand for public transport as a complex system with current states dependent on past states.

The idea of combining CNN for extracting spatial features with LSTM for taking temporal relationships into account has seen a number of successful attempts at fusing those two methods together in one model for predicting ridership [15, 11]. Inclusion of Parallel CNN-LSTM as well as Augmented CNN-LSTM models in the analysis was inspired by Ma et al. [11]. Their research assesses the performance of different prediction methods, including contrasting Sequential and Parallel combinations of the CNN-BiLSTM model. Ma et al. [11] found that a Parallel version of the model performs better than the Sequential counterpart.

Ma et al. [11] use a slightly modified version of LSTM called BiLSTM which stands for Bidirectional Long Short-Term Memory networks - a model that is based on standard LSTM, but it implies passing the input sequence both forward and in reverse. On the conceptual level, this allows the LSTM to be ’aware’ of the whole length of the input sequence, instead of just current and past members at any given time. This approach should potentially lead to better results in pattern recognition and network’s awareness of the context.

There is also a significant difference in authors’ approach towards the CNN portion of the model, specifically in the way that they generate image input for the CNN and the depth of the CNN module of the model. First, Ma et al. [11] attempt to preserve more valid spatial relationship by mapping ridership values onto a matrix and making sure that spatial relationships are preserved on that matrix (relative geographic distance translates into a certain number of empty slots in the resulting image matrix). That way there does not appear to be any information about temporal variation to be available to the CNN module. Lastly, the number of filters that are used in the CNN layers appear very low (16 and 8, or 32 and 16 for another variation).

3. Data

Data access is provided by the Centrum Wiskunde & Informatica (CWI) as part of a research project focusing on building a demand forecasting model for public transport using Machine Learning techniques. The data set contains ridership data aggregated from multiple public transport modes (i.e. metro, bus, train, tram) over the period between 2017-09-18 and 2019-07-21. Ridership for approximately 60,000 5-digit Origin-Destination pairs is present in the original data set.

CWI and other research institutes participating in the project do not have access to the raw data underlying the data set; due to strict privacy rules, data needed to be aggregated to the point where it does not contain data that

could be used to identify individuals. Furthermore in the original study design, spatial resolution was prioritized over temporal resolution; the data set thus contains information about a large number of Origin-Destination (OD) pairs but has limited granularity on the temporal dimension. OD-pairs are constructed based on 5-digit postcodes of origin and destination of the underlying raw data for individual trips.

For each OD-pair, data is aggregated temporally, which in this case implies that per given quarter there are aggregates for months (e.g. ridership in Jan, Feb etc.), day of the week (e.g. aggregated ridership on Mondays, Tuesdays etc. in Q4-2017), hour of the day (e.g. aggregated ridership for 1-2pm time interval on a workday in Q1-2018). One of the caveats of strict privacy rules is that values for ridership that are lower than 300 are assigned into bins ('1-50', '51-100', '101-200', '201-300'). All the values above 300 are rounded to the nearest 20 (e.g. 1605 will be rounded to 1600, 1631 will be rounded to 1640).

Figure 1 shows a visualisation of the values. Rows represent OD-pairs (5-digit origin postcode: 5-digit destination postcode), and columns represent daily ridership values for a given OD-pair.

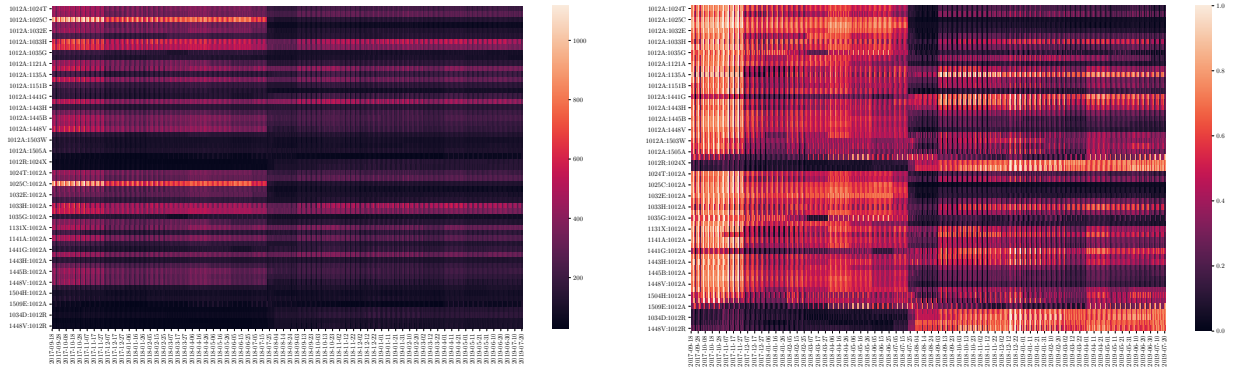

Fig. 1. (a) Raw input matrix; (b) Scaled input matrix.

As can be seen in the raw input matrix in Figure 1 (a), there are significant differences between ridership across OD pairs, with a few having an order of magnitude higher numbers than the rest. To address this, MinMax scaling was used, as implemented in Pedregosa et al. [14]. Each OD-pair (rows) has been scaled independently to ensure that specific demand patterns for a particular route remain intact. The scaled input matrix is depicted in Figure 1 (b).

There are a few reasons for using Min-Max scaling on an interval of (0, 1) as opposed to standardisation. First is that ML algorithms rely on input being positive and preferably normalised, which speeds up convergence and thus reducing training time. Second reason is that scaling is less sensitive to outliers in the data, since it does not rely on the mean for a given OD pair.

4. Methodology

4.1. Description of "image input" for CNN

CNN models are tailored to work with image input data. Normally, images are represented as a matrix of data corresponding to the intensity of color in a given pixel. There is a time-series for each of the OD pairs that is available, combined in the same matrix (with OD pair labels on the Y axis and time on the X axis). Data is then mapped to an interval from 0 to 1 using Min-Max scaling technique.

Due to the nature of the convolutions that CNN performs, even spatial relationships that are not bound together should still be identified, however this might require a bigger (deeper) model and/or more time to train. It is thus better to make spatio-temporal patterns more apparent. Thus it is important to note that ordering of the OD pairs (rows of the resulting matrix) has been adjusted to better reflect the spatial relationships between clusters of trips originating from the same area or having similar destinations. The way that was achieved is through ordering first on the basis of origin, and within the same origin, based on destination. That way, trips that are within the same 4-digit postcode are

clustered together as well as trips to and from the same 5-digit postcode are close together. Figure 1 (b) contains the visualisation of the resulting matrix that is ready to be split into training, validation and test sample sets.

4.2. Example generation and assembling training, validation and test sets

A sliding time-window of 7 days was used over the whole data matrix. That way a set of examples had been produced that contained 6 column vectors (inputs) in the sample and 1 column vector for the day following that period (expected output). Effectively that corresponds to using 6 days worth of data to predict the following day. This set of examples is then partitioned into training, validation and test sets with 80%, 10% and 10% of the available examples assigned to respective sets. The training set was used for training, the validation set was used in the process of tweaking the CNN model to produce better results, while the test set was used to assess performance.

4.3. Neural network model architecture

A summary of model architectures trained using the described example set is presented in Table 1. All hyper-parameters have been fixed in order to assess the relative performance between different models. The 'Adam' learning algorithm had been used, with the learning rate of 0.001.

Table 1. Neural network architecture summary

Model	Input Description	Architectural choices
CNN	59x6x1	Two Convolutional layers (128, then 64 3 by 3 filters respectively) each followed by Max-Pool layers (2 by 2, with a stride of 2). "Same" convolutions.
LSTM	1x354	Each example is directly connected to the the LSTM layer which contains 354 LSTM neurons, that produce a sequence of 354 activation values which are Fully Connected to an output layer with 59 ordinary neurons with ReLu activation function.
Sequential CNN-LSTM	59x6x1	CNN part of that model is exactly the same as in Pure CNN model, however the output of the Max-Pooling layer is flattened into 896 features (14 * 1 * 64), reshaped into (1, 896) to be a valid input for 896 LSTM neurons as a sequence. Output of the LSTM layer is consequently Fully Connected to the 59 Neuron output layer as previously.
Parallel CNN-LSTM	59x6x1 and 1x354	Output of exactly the same Pure CNN and Pure LSTM models had been flattened and then concatenated into one vector of 1250 features (896 and 354 from CNN and LSTM respectively) that is then Fully Connected to the output layer with 59 Neurons.
Augmented Sequential CNN-LSTM	59x6x1 and 1x354	Having Pure LSTM connected after CNN, then, may limit the potential of the LSTM layer of capturing time-series features in the sequence. Thus raw input was concatenated to the feature vector produced by the Pure CNN, in an attempt to provide LSTM layer with extra information. This augmentation process could also be used to input different kind of data into the model at this stage (e.g. exogenous variables like weather, flags for special events etc.).

All the models discussed further were evaluated on the same metric, Mean Squared Error (MSE) as calculated in the following formula: $MSE = \frac{1}{n} * \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$, where n is number of samples in a set, y_i is the actual normalised ridership value to be predicted, $\hat{f}$ is the set of procedures that a given model performs on input data x_i to produce a prediction $\hat{f}(x_i)$. Previous studies [10, 11] have used this loss function for training their models, and it appears to match the context of predicting the continuous variable of ridership.

4.4. Ordinary Least Squares regression

A Fixed Effects Ordinary Least Squares (OLS) model for Panel Data had been used and it is introduced to serve as a baseline for assessing how well Neural Network (NN) models perform. Data from the test period (used for

training NN models) was used for estimating OLS coefficients. The same normalisation method had been employed as previously. Six lagged ridership sequences were used as input variables to predict the following period. By obtaining OLS predictions for inputs in a test period, MSE could be calculated in the same way as for NN models.

5. Results

5.1. Mean Squared Error and Accuracy

Mean Squared Error evaluation values for all models are summarized in Table 2. It appears that Sequential CNN-LSTM converges to approximately the same loss value as Augmented Sequential CNN-LSTM over the course of 1250 epochs, and achieves the lowest MSE_{train} on a training set. LSTM, CNN and Parallel CNN-LSTM perform worse, respectively from lower to higher MSE_{train}.

Table 2. Evaluation values summary

	MSE_{train}	MSE_{test}	$\frac{MSE_{train}}{MSE_{test}}$	Accuracy after training
CNN	0.018105	0.010711	0.591604	0.188
LSTM	0.006634	0.006891	1.038736	0.801
Sequential CNN-LSTM	0.000294	0.000934	3.180420	0.833
Parallel CNN-LSTM	0.024177	0.015674	0.648315	0.188
Augmented Sequential CNN-LSTM	0.000263	0.001089	4.148850	0.892
OLS	-	0.010186	-	

Sequential CNN-LSTM, Pure LSTM as well as Augmented Sequential CNN-LSTM appear to reach accuracy of above 80%. Accuracy of both Pure CNN and Parallel CNN-LSTM did not exceed 20% accuracy throughout 1250 epochs of training.

Sequential CNN-LSTM and Augmented Sequential CNN-LSTM models show at least 6-fold reduction in MSE to the second best model (Pure LSTM). On the other end, the performance of both Pure CNN and Parallel CNN-LSTM on a test set is worse than OLS model results (Table 2).

The ratio MSE_{test}/MSE_{train} emphasises the relative difference between training set and test set performance, which might suggest which direction is more productive to follow when improving the model. This will be explored in more detail in the Discussion section.

5.2. Prediction

Figure 2 shows a cross-sectional view of predictions for all OD pairs for a given day in the test set, and Figure 3 shows a time-series view of predictions for the whole test period for a given OD pair.

In the cross-sectional view in Figure 2 it can be seen that models with lower MSE on the test set predict ridership values that are closer to ground truth. Even though it seems like Pure CNN predictions fit the actual values well, assessing it from time-series view of predictions (Figure 3 for a given OD pair reveals that CNN, in fact, predicts values in a small proximity to the average ridership of a given OD pair. This implies that cross-sectional view for Pure CNN performance varies depending how close ridership is to the average for any OD pair. While predictions of the LSTM model seem to be accurate most of the time, for one particular OD pair the prediction is consistently null.

6. Discussion

It appears that Sequential CNN-LSTM is the best model in terms of performance on the test set. Its score, however, is quite close to that of the suggested Augmented CNN-LSTM model. Moreover, in terms of convergence speed, the Sequential CNN-LSTM model is the fastest in terms of passes (epochs) required to fit the training set.

This suggests that features extracted by the CNN module within the Sequential CNN-LSTM model are sufficient for forecasting ridership accurately. Given that the Pure CNN did not perform as well on its own, it is likely that the LSTM module on top of CNN had played a crucial role in model performance.

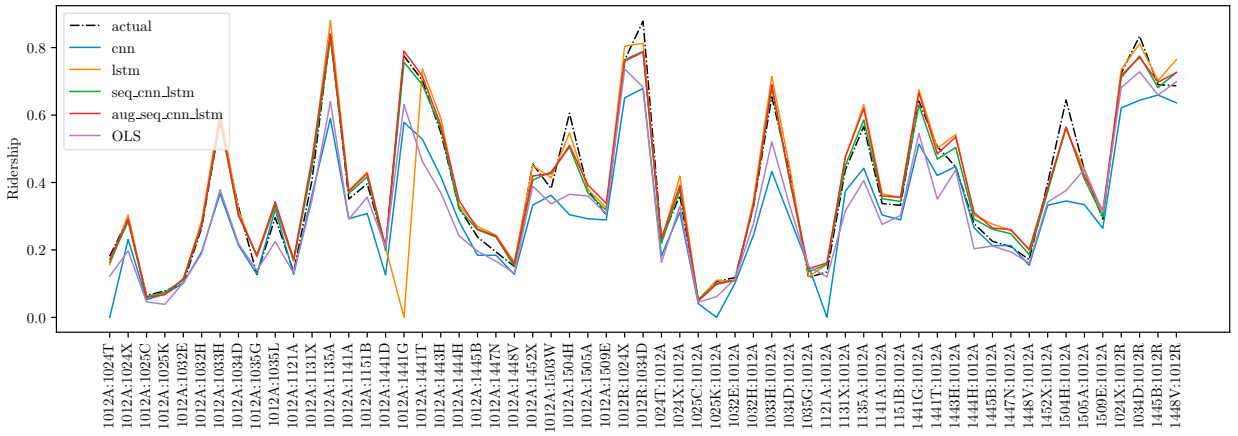

Fig. 2. Prediction for 2019-06-27 (cross-section view)

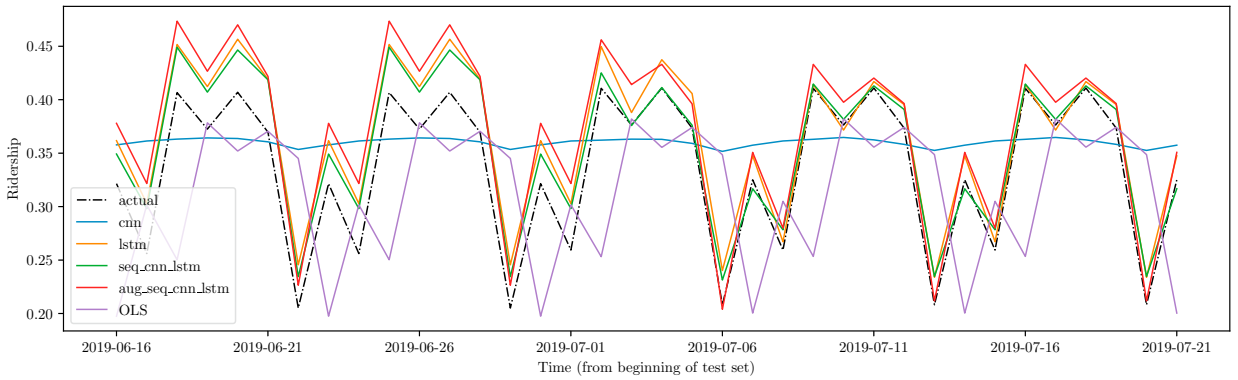

Fig. 3. Prediction for 1012A:1131X OD pair, whole test period (time-series view)

While Augmented CNN-LSTM had achieved similar results to its Sequential counterpart, additional raw input does not seem to result in lower MSE. Also, in case the input sequence was larger (e.g. the past 14 days worth of data, or if the time-frame was 10 minutes instead of 1 day), then the concatenated input would require a much larger LSTM layer to be trained, which does not scale linearly in terms of compute time. That said, it appears the augmentation method itself works, and potentially could be utilised to insert exogenous variables or metadata about a given example into the model (e.g. weather data, features of a given example etc.).

Ratios between training and test loss shows that models with better performance have seen a relatively larger ratio of MSE_{test}/MSE_{train} than poorer performing models. Ratio of lower than one implies that Pure CNN and Parallel CNN-LSTM are not able to fit the training well and could potentially be improved by increasing model complexity and/or further hyper-parameter tuning.

Models with exceptional performance, in contrast, seem to over-fit the training set, since MSE_{test} is 3 to 4 times higher than MSE_{train} for Sequential and Augmented CNN-LSTM models respectively. This could be potentially mitigated by including Batch Normalisation for instance in the model, or a technique called ”Early Stopping”, where performance on a validation set is checked after every epoch and in case of stagnation for a specified number of epochs (or in case the loss on validation set goes up), then training is stopped and the model is prevented from over-fitting the training set.

Interestingly, Parallel CNN-LSTM did not perform as well as expected from the findings of Ma et al. [11]. Its performance had been closer to that of Pure CNN. Given that the output layer is a simple Fully Connected layer with 59 neurons, it appears that the fact that there are almost twice the amount of features coming from the CNN part than

that of LSTM steers the performance of the model towards that of Pure CNN. It is possible that the results would be better if there had been a deeper NN on top of the Concatenated layer, which could distinguish between CNN and LSTM feature sets and utilise them more efficiently.

The prediction pattern of the OLS model in the time series view (Figure 3) reveals that the forecast of the OLS model is equivalent to that of a naive model forecasting the next day to be equal to the last known previous day. However, the OLS performance on the test set is better than both the Pure CNN model and the Parallel CNN-LSTM. This implies that the naive model would be better than those two as well. Nevertheless, it is possible that given more complexity in the input data, Pure CNN and Parallel CNN-LSTM could still outperform OLS considering their theoretical capacity for learning complex non-linear relationships in the data.

Pure CNN does not appear to distinguish temporal relationships in the input data as well as theory would imply, since it is clear that for a given OD pair CNN predicts ridership value that minimises the loss function (MSE) for a given OD pair (Figure 3). Cross-sectional view, however, indicates that spatial features are well incorporated in the output of the CNN. It is possible that such outcome is the result of the low resolution of the input, more so in the temporal dimension, since Ma et al. [10] shows that it is indeed possible to train a reliable forecasting Pure CNN model using granular spatio-temporal data.

Visual inspection of predictions suggests that the LSTM model is sophisticated enough to produce accurate forecasts for the majority of OD pairs. It is not clear why ”1012A:1441G” in particular is consistently forecast to have 0 ridership by Pure LSTM. Yet, performance of the Pure LSTM on the other OD pairs and improvement in performance from Pure CNN to Sequential CNN-LSTM suggests that there is a lot of potential for building accurate forecasting models using multiple data sources feeding into an LSTM layer.

References

[1] Caceres, H., Batta, R., He, Q., 2017. School bus routing with stochastic demand and duration constraints. Transportation Science 51, 1349–1364. URL: https://doi.org/10.1287/trsc.2016.0721, doi:10.1287/trsc.2016.0721.

[2] Franklin, J.A., 2006. Recurrent neural networks for music computation. INFORMS Journal on Computing 18, 321–338. URL: https://doi.org/10.1287/ijoc.1050.0131, doi:10.1287/ijoc.1050.0131.

[3] Jin, J.G., Teo, K.M., Odoni, A.R., 2016. Optimizing bus bridging services in response to disruptions of urban transit rail networks. Transportation Science 50, 790–804. URL: https://doi.org/10.1287/trsc.2014.0577, doi:10.1287/trsc.2014.0577.

[4] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732.

[5] Knoppers, P., Muller, T., 1995. Optimized transfer opportunities in public transport. Transportation Science 29, 101–105. URL: https://doi.org/10.1287/trsc.29.1.101, doi:10.1287/trsc.29.1.101.

[6] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. Communications of the ACM 60, 84 – 90. URL: https://search-ebscohost-com.proxy.uba.uva.nl:2443/login.aspx?direct=true&db=buh&AN=123446102&site=ehost-live&scope=site.

[7] Lawrence, S., Giles, C.L., Ah Chung Tsoi, Back, A.D., 1997. Face recognition: a convolutional neural-network approach. IEEE Transactions on Neural Networks 8, 98–113.

[8] LeCun, Y., Bengio, Y., et al., 1998. Convolutional networks for images, speech, and time series, the handbook of brain theory and neural networks.

[9] Li, T., Chan, A.B., Chun, A.H., 2010. Automatic musical pattern feature extraction using convolutional neural network. Genre 10, 1x1.

[10] Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., Wang, Y., 2017. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. Sensors (Basel, Switzerland) 17. URL: https://doaj.org/article/f9eb31dbbc9f459da710293c39ddc03f.

[11] Ma, X., Zhang, J., Du, B., Ding, C., Sun, L., 2019. Parallel architecture of convolutional bi-directional lstm neural networks for network-wide metro ridership prediction. IEEE Transactions on Intelligent Transportation Systems 20, 2278–2288.

[12] Oquab, M., Bottou, L., Laptev, I., Sivic, J., 2014. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks, in: IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, United States. URL: https://hal.inria.fr/hal-00911179. conference version of the paper.

[13] de Oña, J., de Oña, R., 2015. Quality of service in public transport based on customer satisfaction surveys: A review and assessment of methodological approaches. Transportation Science 49, 605–622. URL: https://doi.org/10.1287/trsc.2014.0544, doi:10.1287/trsc.2014.0544.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[15] Zhang, J., Ma, X., Ding, C., Wang, Y., Liu, J., 2018. Forecasting subway demand in large-scale networks: a deep learning approach. Technical Report.