Multi-Task Learning Based Neural Networks for Density Estimation in City Environment

Mostafa Karimzadeh, Samuel Martin Schwegler, Torsten Braun
Institute of Computer Science, University of Bern, Switzerland
Email : {mostafa.karimzadeh, torsten.braun}@inf.unibe.ch, samuel.schwegler@students.unibe.ch

***Abstract*—abstract goes here, by Mostafa**

***Index Terms*—Mobile analysis, Mobility Prediction, Mobility Behavior, Location based Services.**

I. Introduction

introduction goes here by Mostafa

II. Related Work

related work goes here by Mostafa

III. System Model

system model goes here by Samuel

A density predictor attempts to estimate the number objects in a given area. In this work we used On Board Units (OBUs) in a Vehicular Ad Hoc Network (VANet) in the urban area of Porto, Portugal. The OBU devices were installed on vehicles such as busses.

The mobility data comes as a vector in the shape $x_t = [x_t^1, x_t^2, ..., x_t^N]$ where t is the current time step and x^n is the sum of connected OBUs to one of b Road-Side-Units (RSUs).

S2 is a library for spherical geometry. The S2 library enables spatial indexing and approximating regions as collection of discrete "S2 cells". This is done by working exclusively with spherical projections, this mapping enables to approximate the entire Earth's surface with a minimum distortion of 0.56%. [1]. The globe starts as a cube with six cells. There are 31 cell levels existing, each level divides the parent cells into four smaller cells. The smallest cell level contains cells covering an area of 1^2cm. For every longitude latitude position on the planet and a level the S2 library can return the unique cell which contains this coordinates. The unique cell labeling is achieved by using a Hilbert curve on each of the level 0 cells. This feature is used to group the RSUs into collections. In Fig. 1 the city centre of Porto can be seen with RSUs (red dots) and S2 cells of level 13 (blue borders).

For the density prediction the sum of OBUs connected to RSUs is mapped into S2 cells. This helps keeping the spatial context and enables decreasing the sparsity of input data. By only looking at single RSUs there are many phases were zero or nearly zero OBUs are connected. The input data is sparse if many of its coefficients are zero. The grouping also generalizes the data and decreases the arbitrariness of the possibility for an OBU to connect to a RSU.

The goal of our work was to make a prediction for the mobility data vector at the time $t+1$ depending on the state

Fig. 1: city centre of Porto with RSUs and S2 Cells. Created using leaflet[2]

in time t. In this case y^n is the sum of connected OBUs to any of r RSUs which belongs to a S2 cell.

The RSUs are not evenly distributed into S2 cells, this can be seen in Fig. 1. For example U RSUs are existing for the first S2 cell and W RSUs for the last of our M S2 cells.

In the temporal dimension the data was also grouped into windows of 30 minutes which also helps decreasing the sparsity. Therefore we have M S2 cells with 48 time-steps t per day as our input dataset.

The grouping in time and space can be seen as:

$$\begin{aligned} & y_{t+1} \\ &= [(u_{t+1}^1 + u_{t+1}^2 + ... + u_{t+1}^U), ..., (w_{t+}^1 + w_{t+1}^2 + ... + w_{t+1}^W)] \\ &= [y_{t+1}^1, ..., y_{t+1}^M] \end{aligned}$$

The minimization of the differences between the ground truth and predicted densities is the goal of this work. We want to achieve this by using Neural Networks (NNs).

A. Neural Networks for Density Prediction

description of the mobility predictor goes here by Samuel

The density of OBUs in a VANET is changing over time. Traffic is not linear and different for each cell. Simple density estimators are not well suited for this problem. NNs have the ability of learning this complex behaviour.

A regular neural network can not distinguish the spatio-temporal correlation. Recurrent Neural Networks (RNNs) can keep their internal memory during the processing of sequential inputs. They jointly explore the spatio-temporal relationships. [3] But simple RNNs can have problems with keeping long-term dependencies. RNNs have a vanishing gradient which causes the gradient to shrink at a rate that is exponential in the number of time steps. To avoid this a Long Short Team Memory (LSTM) is used in this work. A LSTM is a specialised type of a RNN it was first mentioned in 1997 by Sepp Hochreiter and Jürgen Schmidhuber. [4] LSTMs reparameterizing the RNN so their gradient cannot vanish. LSTMs have the ability of keeping information learned over a long period of time. This is achieved by a special designed memory cell. [5] A LSTM can be expressed as

$$m^t = g^t \odot i^{-t} + f^t * m^{t-1}$$
$$h^t = o^t \odot m^t$$

where m is the internal state or memory, h the hidden state and i the input state. The gates handle the information flow of states. g defines whether the input state enters the internal state. f is the forget gate which decides if the internal state should forget the previous internal state. Finally the output gate o decides if the internal state is passed to the output and hidden state of the next time step. $\odot$ denotes element wise multiplication. [6]

These ability leading LSTMs to have an excellent performance for predicting densities by learning temporal dependencies.

As a benchmark for advanced Neural Network architectures a LSTM that is only capable of predicting the density for one S2 cell is used. The input features are only temporal not spatial. This LSTM does not get any additional information from other S2 cells or other predictors. This architecture is referenced in this paper as "single-task". The system model is shown in Fig. 2

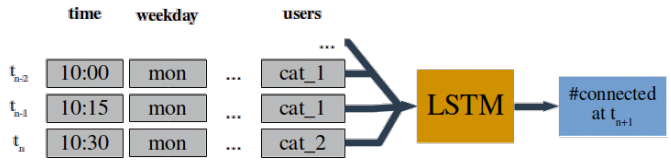

Fig. 2: Predictor for a single task

B. Accelerated Neural Networks for Density Prediction

description of the mobility predictor goes here by Samuel

For advanced predictions the spatio-temporal correlations are used. The input features are grouped spatially by S2 cell. A task in this case is predicting the density for one s2 cell in a given interval of time. The tasks are related to each others because the OBUs are moving between RSUs in multiple S2 cells. The resemblance and dissimilitarity are information to explore. This knowledge helps to improve the predictions. In our work we want to show that sharing information between tasks does not only help to increase the accuracy of the predictor but does also decreases the training and testing time per task.

For this improvements a multi-task architecture is used. This network architecture is capable of sharing information between tasks. Multi-task learning is a learning paradigm in machine learning that aims to reduce the model error of a target task by utilizing related auxiliary tasks. In comparison of Transfer Learning multi-task learning learning simultaneously over several correlated tasks and not after another in sequential. Multi-Task learning means to learn a joint model of all tasks in parallel. [7] For the density prediction all of the tasks are of equal importance. They are all auxiliary tasks for each others, learning one should enhance another task. Therefore transfer learning with a parent providing information to a child process does not cover the needs.

An advantage of multi-task learning is attention focusing. For noisy tasks or limited and high-dimensional data it is difficult for a model to differentiate between relevant and irrelevant features. Here multi-task learning can help the model to focus. Other tasks provide additional evidence whether a feature is relevant or not. It is also possible that some tasks are better in learning tasks than others. With the interaction the model learns to predict important features. Multi-task learning helps also reducing the risk of overfitting by introducing an inductive bias. [8] With multi-task learning the need for data samples for an accurate training and testing can be lowered. [9] In our case we have a limited amount of entries as our dataset and want to take advantage of these points.

In this work different multi-task architectures were used. All of them have a shared learning machine machine which gets inputs from all our M cells together and dedicated learning machines that only serves a particular task for each of the M cells. The output is the concatenation of the shared and a dedicated layer.

In this work we tested architectures with hard and soft parameter sharing. In soft parameter sharing each task has its own model with own parameters. With hard parameter sharing hidden layers are shared between all the parts and then having task specific output layers. [8]

In the first architecture the dedicated learning machines only get the input for their own task, like in the "single-task" architecture. We call this 1-to-n mode. The second architecture provides all inputs to all dedicated layers. This mode we call n-to-n. The differences are shown in Fig. 3. These two architectures are using soft parameter sharing. As a third multi-task architecture a method that uses hard parameter sharing is used. This one is referenced as "hard" in this paper.

In further work additional data like weather which would be the same for all the s2 cells could be applied.

More Content

IV. Evaluation

In the following section an evaluation methodology to validate the proposed density prediction model is presented.

1) Dataset: datasets goes here by Mostafa

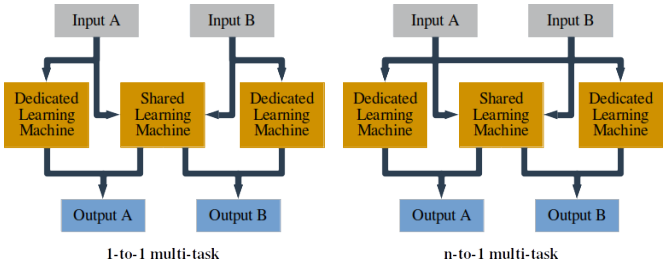

Fig. 3: Multi-Task Architecture (for simplification with only two tasks

2) Evaluation Metrics: evaluation metrics goes here by Samuel Our code is written too handle the density prediction as classification or as regression problem.

The classification problem is a one-of-many Multi-Class Classification. The density of OBUs is mapped into C classes. The ground-truth is a one-hot encoded vector t with one positive class an $C-1$ negative classes. Categorical crossentropy is the loss function for this problem:

$$L(y,\hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{i,j} * log(\hat{y}_{i,j}))$$

Where $\hat{y}$ is our predicted value.

We measured the accuracy, recall and precision of our prediction. To check that recall an precision are balanced we use further the f1-score which is the harmonic mean between recall an precision. The harmonic mean penalizes extreme values.

For regression problems we used the mean squared error (mse) as loss-function:

$$L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Other metrics were root mean squared error (rmse) and mean absolute error (mae)

3) Experimental Settings: evaluation parameters goes here by Samuel

As test/train split we implemented a time-series cross-validation on a rolling basis (also called ”expanding window validation”). The training starts initially on a small set, then the model is tested on one batch. This batch gets afterwards added to the training set for the next batch. So the training set increased for every part. This method is helpful if there is only a limited amount of entries in the data set. For example we had only data for some months and wanted to train and test in different seasons and not only train in winter and test in summer. This would be happened with a 70/30 split. Of course shuffling is not an option with time series data as it would vanish structure.With this form of train and test split data is used for both training and testing and the time series are taken into account. As an alternative also a ”sliding window” test train split was tested. In this case the training section has allways the same length and older values are dropped. An advantage would be the faster training time but the tradeoff is the lost of information.

For running the experiment we took eight connected S2 cells in the city centre of Porto with the level of 13 which covers in average an area of 1.27 km^2. [1]. The code was also applied on an other data set which was not grouped into spatial cells. In this case densities were predicted for RSUs. In this work we will go further into the work on the Porto set.

In the classification mode we mapped the count of connected users into three categories. The boundaries of the categories are chosen in a way that every category contains the same amount of elements. So no category is over represented. Because we avoided imbalanced class distribution accuracy is a good way to check our performance.

The batch size of our LSTM was set to 12, with 70 neurons for dedicated layers. We took time intervals of 30 minutes for the density and worked with a time span of six months. 30 minutes were optimal in our case because within this timespan the sparsity of the input data is reduced and it is still granular enough to see patterns over the course of the day. The used dataset contained tracefiles for 1000 users in a VANET. The data was collected from winter to early summer 2017. For recognizing seasonal patterns it would be of course better to have data over multiple years.

For the machine learning we used the keras library[10] which is a high-level neural network API, written in Python. In our case keras war running on top of tensor flow. In order to get the multi-task learning the functional API was used.

More Content

V. Evaluation Results

1) Density Prediction Results: results goes here by Samuel In this section we show how multi-task learning improved the performance of density prediction in urban areas.

To get reliable results we made 100 runs and then took the average of our performances. For comparison the single non multi-task architecture is showed with the 1ton and nton multi-task that are useing soft parameter sharing and the hard mutli-taks method that uses hard parameter sharing. The used cells are in the city centre of porto. The density data was from a tuesday.

An example for predictions in the regression mode are shown in Fig. 4 the density was mapped from 0 (lowest density) to 1 (maximal density).

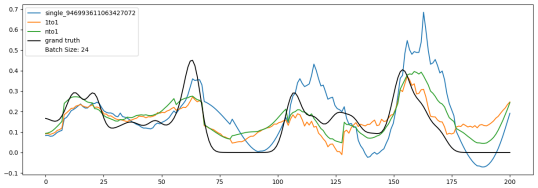

Fig. 4: compared predictions for one example cell

In this example can be seen that the multi-task architectures are reacting less extreme in comparison to the ”single” model.

The main focus of the prediction laid on a classification. The biggest improvement of performance with multi-task architectures can be seen in the f1 score (Fig. 5).

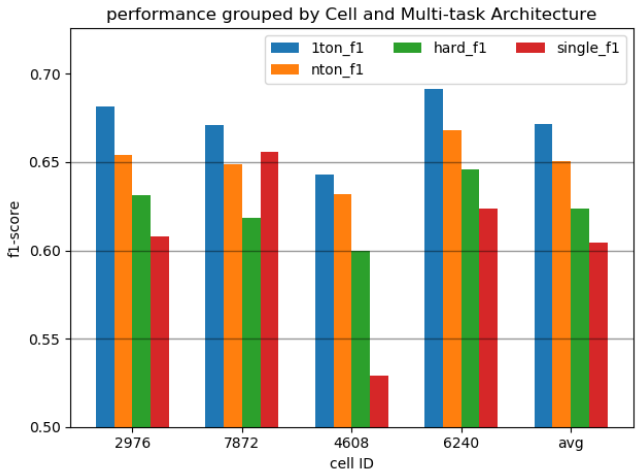

Fig. 5: f1-score for different cells and multi-task modes

The accuracy was slightly improved with multi-task methods that were using soft parameter sharing. But the hard-parameter sharing approach lowered the average performance form a single approach by 4.4% (Fig. 6).

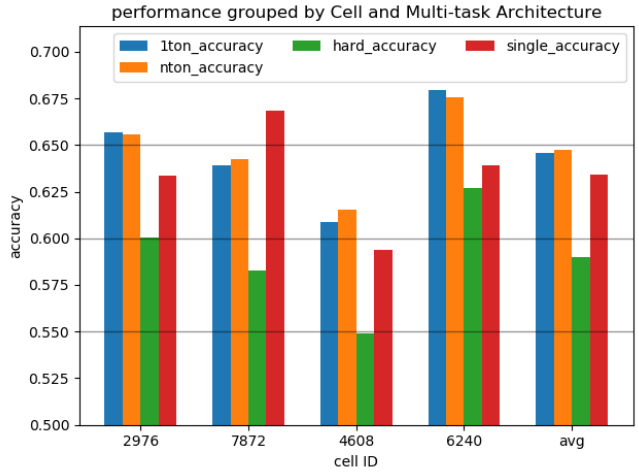

Fig. 6: accuracy for different cells and multi-task modes

A high improvement the running time. We calculated the runtime for multi-task predictions by dividing the overall runtime by the number of cells. In this chart 100% is the running time of the single architecture. The 1ton multi-task architecture finishes after 77.5% of the single time (Fig. 7).

The differences between the two soft-parameter multi-task approaches are small, we can not say in general that one outperforms an other. But multi-task approaches are performing better than non multi-task approaches and soft-parameter sharing is better than hard-parameter sharing for our problem.

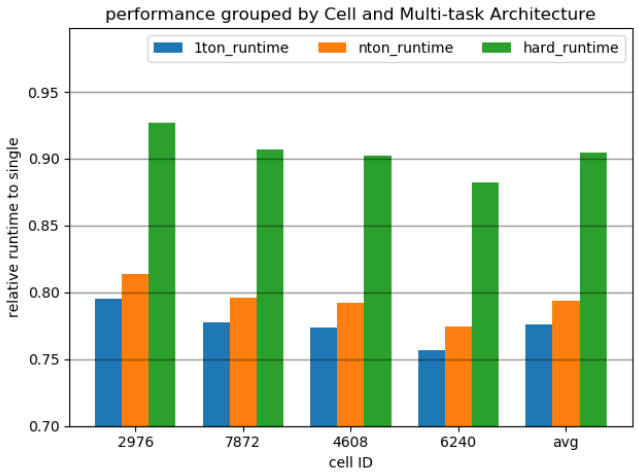

Fig. 7: relative runtime for different cells and multi-task modes. The runtime of the single mode is 1

VI. CONCLUSIONS

conclusion goes here by Mostafa

REFERENCES

[1] s2geometry io@googlegroups.com. S2 cell statistics. [Online]. Available: https://s2geometry.io/resources/s2cell_statistics.html

[2] leaflet. [Online]. Available: https://leafletjs.com/

[3] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, "Spatio-temporal wireless traffic prediction with recurrent neural network," *IEEE Wireless Communications Letters*, vol. 7, no. 4, pp. 554–557, Aug 2018.

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

[5] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," *Journal of Machine Learning Research*, 2015.

[6] H. Wang, B. Raj, and E. P. Xing, "On the origin of deep learning," *CoRR*, vol. abs/1702.07800, pp. 47–52, 2017. [Online]. Available: http://arxiv.org/abs/1702.07800

[7] S. Spieckermann, "Multi-task and transfer learning with recurrent neural networks," Dissertation, Technische Universität München, München, 2015.

[8] S. Ruder, "An overview of multi-task learning in deep neural networks," *CoRR*, vol. abs/1706.05098, 2017. [Online]. Available: http://arxiv.org/abs/1706.05098

[9] Y. Zhang and Q. Yang, "A survey on multi-task learning," 2017.

[10] F. Chollet *et al.*, "Keras," https://keras.io, 2015.