CROWDEDNESS PREDICTION IN PUBLIC TRANSPORT UNDER COVID-19

Tim Kaiser (s201792), Yevhen Khavrona (s201441), Jan Cedric Issel (s201232)

1. INTRODUCTION

During Covid-19 pandemic, social distancing in public transport is an important matter to prevent spreading the virus. Thus, it would be beneficial to know when and where there are bottlenecks in the public transport network. Our goal is to reduce this capacity problem by predicting the crowdedness of bus lines for a specified time interval. In our project we focus on the bus line '150S' under the assumption that our results can be scaled to other bus lines. As a data basis, we use measurements of the passenger numbers that enter and leave buses on the line. This information on the one hand stems from check-in/check-out numbers of Rejsekort users in the respective bus line. On the other hand, a small fraction of the buses has been equipped by the operator company Movia with sensor-based passenger counters that provide this information as well. To evaluate how to achieve the best possible prediction accuracy, we test different machine learning model architectures and different combinations of the available data. The source code of this project can be found at https://github.com/cdrc1103/DL_Project13e.

2. DATA ANALYSIS

In the beginning of our data analysis, we decided to focus on the passenger numbers measured by the Rejsekort check-ins / check-outs since data was available for a longer time frame here. The measured passenger numbers were given in an array of shape: (5786, 10, 10). This array provides a mapping of the number of passengers that enter the bus line '150S' at a certain timing point and leave it at another timing point. A timing point aggregates a small number of neighboring bus stops in order to reduce the dimensionality of the problem. The measurements are summed up per hour giving us 5786 data points. Thus, we have aggregated information about the journeys of passengers traveling with bus line '150S' that is visualized as a heatmap in Fig. 1. As can be seen, there is a clear trend in the entering and leaving behavior of the passengers. Judging by the heatmap, passengers enter or leave the bus mostly at specific timing points like Nørreport St. (NPST1), Klampenborgvej (KLBV) or Hans Knudsens Plads (HKP). Also, patterns like night hours and rush hours are present in the data as can be seen in Fig. 2 for the example of people entering at Nørreport St. and leaving at Nærum St. This lead us to the conclusion that these patterns could be suitable features for our prediction.

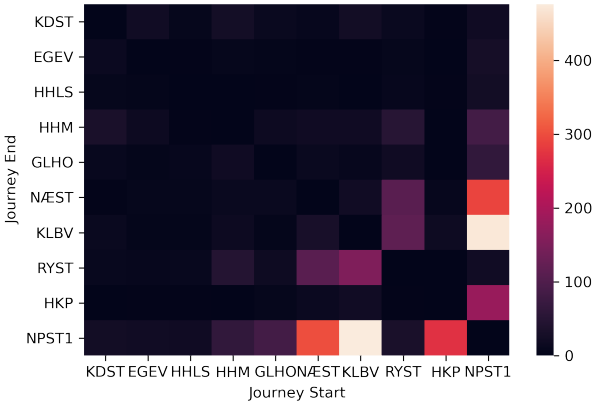

Fig. 1. Heatmap of the number of passengers entering and leaving at specific timing points within 48 hours.

However, we wish to predict the crowdedness in a bus line between two timing points. The initial configuration of the data doesn't provide this information. Therefore, we created nine intervals between the ten timing points that aggregate all passengers that are currently in the bus according to the timing point mapping. Since the bus line drives in both directions we created two arrays each with the shape: (5786, 9). Thus, we received two multivariate time series of 5786 hours at nine different intervals. An excerpt of this time series can be seen in Fig. 3 which shows common and different patterns for the respective intervals depending on the time on the day of the week and the hour of the day. For instance, during the night at around 00:00 all three intervals show no significant passenger numbers. On the other hand, all three intervals show two peaks of varying extent that are related to standard rush hours. Also, the first March 2020 was a Sunday, which is why the overall passenger number is lower on this day. Based on these findings, we decided to use approximate time intervals like morning, afternoon, evening and night as features for the prediction of the passenger numbers.

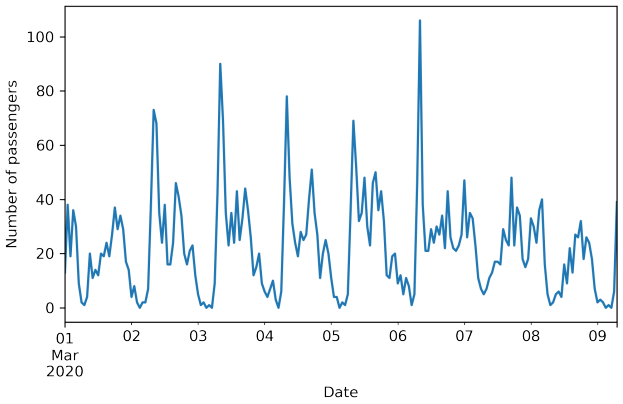

Fig. 2. Time series of passengers entering at Nørreport St. and leaving at Nærum St. within 200 hours.

3. APPROACH 1: MULTISTEP LSTM

From our preprocessing steps in Section 2 we obtained a time series. In a time series each data point can be affected by the values of previous and future data points. In our case we have a multivariate time series of nine different intervals that affect each other as well. From the lecture we knew that this sequential data structure can be modeled effectively with recurrent neural networks. However, a classic RNN is limited by the problem of vanishing gradients for long sequences. Thus, our initial approach was to apply the standard LSTM from the lecture to our data.

To limit the necessary training time of the model, we decided to use only one of the two arrays that we created initially (the number of passengers in buses that start at Nørreport St.). For the real life application, we figured that a prediction of the passenger numbers in the upcoming 24 hours would be beneficial to avoid capacity bottlenecks. However, the classic LSTM is used to predict only the next element of the sequence, i.e. the next hour. We decided to call the model repeatedly while incrementing the sequence position every time in order to receive a sequence of 24 predictions. For the training data, we chose a sequence length of 72 hours. Thus, we split our initial time series into training sequences of 71 data points while the 72nd point is subject to the prediction. For the model evaluation during and after training, we used 10% of the data set each. The rest was available for training. As complexity controlling parameters we set the number of hidden dimensions to 30 and the number of layers to 3. As optimizer we used Adam with a learning rate of 0.001 without regularization. We trained the model for 35 epochs. In Fig. 4, predictions of the next 24 hours starting from the 160th hour and their corresponding true values are visualized. Obviously, our model underfits the data significantly. We tested several hyperparameter configurations and didn't achieve any improvements in the accuracy. Our guess for the poor performance of the model would be that this architecture is not optimal for predicting sequences. Thus, we reasoned that we have to find a model architecture that is able to successfully map between sequences of varying length.

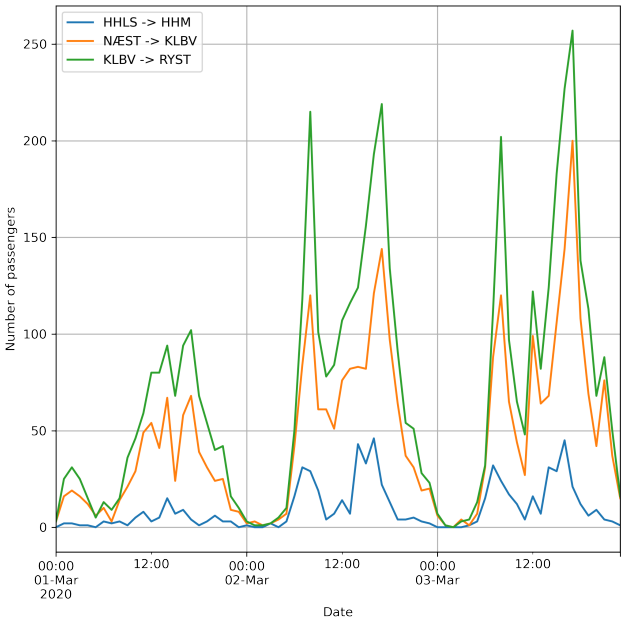

Fig. 3. Number of passengers for three of nine intervals within a 72-hour time interval.

4. APPROACH 2: SEQ2SEQ MODEL

In order to create a model that is able to generate the predictions that we need, we turned our attention to Seq2Seq models that are based on encoder-decoder architecture. These models are able to take a sequence of a specific length as an input and generate a sequence of a different length as an output. In general, a simple encoder-decoder model works by first encoding the input sequence into a so-called context vector. Then this vector is used by the model's decoder in the decoding phase to produce outputs. Thus, the characteristics of Seq2Seq models make them particularly suitable for our task - generating an output sequence from a larger input sequence.

We used the same nine intervals as a basis for the model's input. To capture temporal features of the sequential crowdedness data, we added two new features to the data - 'time of the day' and 'day of the week'. The former represents part of the day in which a particular bus journey started (0 for morning, 1 for afternoon, 2 for evening and 3 for night) while the latter simply encodes the number of the day of the week the journey took place on (0 for Monday, 1 for Tuesday etc.). The exact time frames of those 4 parts of a day were determined arbitrarily based on the daily crowdedness patterns observed on line '150S' (e.g. we defined 'morning' as part of a day

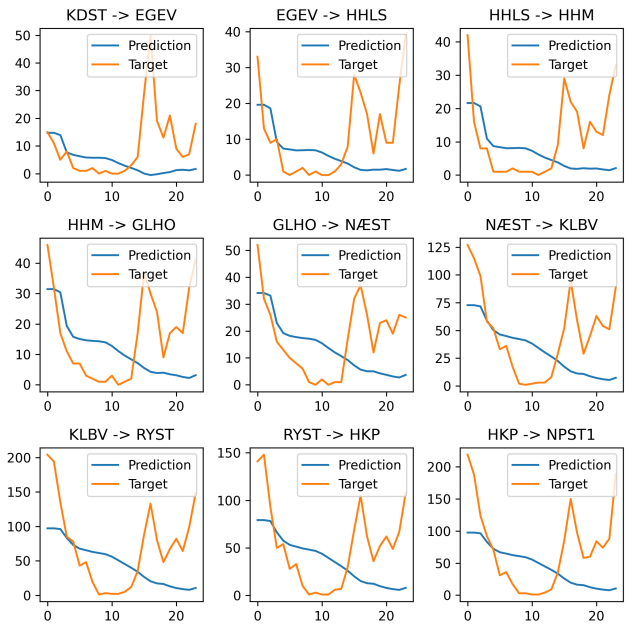

Fig. 4. Prediction and true value of the passenger numbers for the next 24 hours.

that starts at 5:00 AM and ends at 11:59 AM). These temporal features can be directly fed into our Seq2Seq model as they themselves represent periodic time series. There were other approaches we could resort to in order to take into account those temporal features, such as feeding them into a separate dedicated LSTM cell and merging the cell's output with the main model's encoder outputs. However, we decided to keep this part simple due to the time limits of our project. All the features were scaled then by applying the min-max normalisation (scaling) to speed up the model's convergence.

Since Seq2Seq models work with sequences, we had to split our dataset of 5786 samples (each consisting of data for 9 intervals) into a bunch of smaller subsequences. We decided to train our model on 168 hours (7 days) of past crowdedness data and predict crowdedness for the next 48 hours (2 days). Thus, the total length of each subsequence consisting of an input to the model and a label is 216 (hours). To split the data in such subsequences, we applied the sliding window approach: in each iteration the 'window' was sliding forward by one hour and the resulting sequence comprising last 216 hours was added to the list of input sequences. In total, 5570 sequences were generated.

This data is then fed into our encoder-decoder model. The encoder is a simple LSTM cell that accepts sequences of length 168, each consisting of 11 features. The last hidden state of the encoder is used as the first hidden state of the decoder. The decoder itself is also built around a single LSTM cell. In order to let the model concentrate more on important parts of the demand sequence (and ignore random spikes in demand, for instance), we decided to add an attention layer between the encoder and decoder. We based our implementation of the layer on the one proposed by Dzmitry Bahdanau in the paper that introduced the concept of attention [1]. The attention layer of our model takes the output of the encoder (of length 168) and combines it with each new hidden state of the decoder. Then, a vector of attention weights is calculated by feeding the combined outputs through a linear layer and applying the softmax activation function to the result. The attention weights are combined with the outputs of the encoder (into a context vector) to signal which parts of the demand sequence should be deemed more important. The decoder repeatedly produces an output by receiving an input vector with 11 features, combining it with the context vector and passing the result to the decoder's LSTM cell. The produced output is used as an input for the next prediction round. As the initial input to the decoder, which signals the start of the output sequence, we use a vector of ones. Altogether, the decoder generates 48 predictions that represent the 48-hour prediction window.

Since one of the most important goals of our project is to come up with a scalable solution for bus crowdedness prediction, the training time of our model has to be kept within reasonable limits. Thus, we decided to limit the number of training samples the model sees during training phase to 120 out of 5570 available. We randomise the selection of the training sample in order to reduce dependence of the data on particular short-term demand trends. To speed up training even more, we use a single optimiser for the combined encoder-decoder model instead of using several dedicated optimisers for each of the submodels. As a consequence, training the model for 100 epochs with training batch size of 10 takes only several minutes with GPUs available on Google Colab. During the testing phase the model generated predictions that were close to the actual demand, meaning that the model was able to catch the demand pattern well. An example of predicted demand is presented in Fig. 5. The graph shows how the model accurately represents the general demand pattern on Saturday-Sunday in the interval between GLHO and NAEST timing points and smoothes out irregularities in the demand. It's important to stress the fact that in the training phase the model could only see the data from one week prior to the prediction's time frame.

Further experiments with more LSTM layers and bidirectional LSTM both in the encoder and decoder did not lead to significant increase in accuracy of generated predictions. On the contrary, the model started overfitting due to the small size of the training set and overly complex model. Since the time needed to train such a model increased drastically, we decided that our initial model achieves an acceptable balance between the time needed for training and accuracy of predictions.

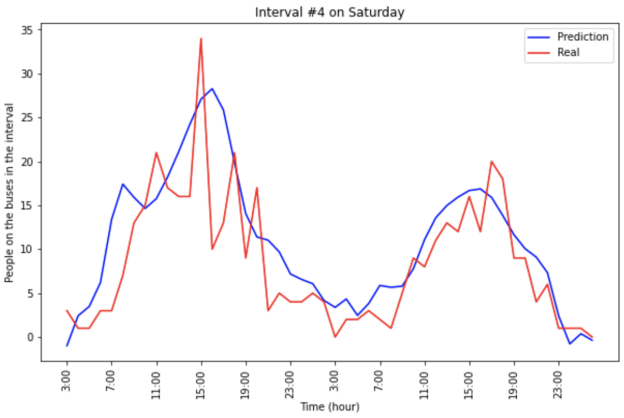

Fig. 5. An example of Seq2Seq model's prediction of line '150S' crowdedness on Saturday-Sunday in the interval between GLHO and NAEST timing points.

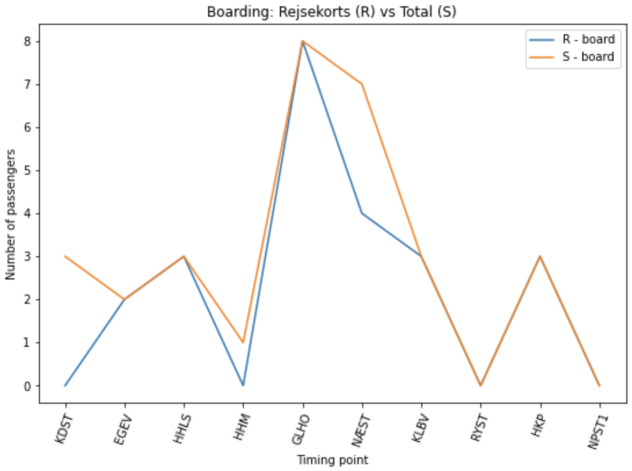

Fig. 6. The orange graph shows how many people entered the bus in total at each timing point. The blue graph shows how many of these were Rejsekort users.

5. REAL CROWDEDNESS PREDICTION

Our network is now capable of predicting the number of Rejsekort users travelling with bus line '150S' within the next couple of hours. This however, does not describe the real crowdedness in the respective bus. Due to different payment systems, there can be more passengers on a bus than the Rejsekort system registered. Some people travel using alternate means of pay like mobile tickets or monthly passes. This needs to be accounted for in order to get the most accurate prediction of the number of passengers in a bus. The problem here is that there is not a constant percentage of Rejsekort users among the passengers. The exact amount varies depending on the time of the day and the day of the week. Thankfully, we were provided with another data set that contained the actual number of passengers for a few selected buses (the ones equipped with sensors). This data set was collected using the mentioned sensors mounted above the entrances and exits. However, due to the small number of buses equipped with this technology the data set was too small to directly use it for a reliable prediction model that doesn't overfit on the training data. Therefore, we had to train a standard feedforward network to predict the real number of passengers based on the Rejsekort data. After the integration of the sensor data, we were also able to apply the real passenger number predictions to our previous results. We tested our network by predicting the real passenger numbers for the next 48 hours based on the respective Rejsekort data. Fig. 6 shows an example of difference between the number of passengers travelling with Rejsekort and the total number of passengers as counted by the on-bus sensor.

For the architecture of the model, we chose a simple feedforward neural network. Since the model yielded satisfying results, we stuck with this model. Our architecture uses three layers. The first layer has an input dimension of eleven. Nine of the input values are the mentioned timing point intervals on the '150S' bus line. Additionally, there is an input parameter that specifies the time of the day (morning, afternoon, evening or night) and one for the part of the week (weekday or weekend). This allows us to take weekly and daily patterns into account. The next layer is another linear layer with an input size of 40. The final (output) layer takes an input of size 30 and produces an output of nine which corresponds to one value for every interval on the line. As activation function, we apply ReLu between the layers. The network was trained for 100 epochs, even though it already showed signs of convergence after just a couple of epochs. Fig. 7 shows several examples of predicted total crowdedness. The biggest issue overall was the small size of the sensor dataset. As sensors are installed on a fraction of Movia's buses, there were only around 500 bus rides with sensors on the line '150S' in the four-month period between August and November. Thus, for some combinations of 'time of the day' and 'day of the week' features there are only several data points available which naturally leads to less accurate predictions for such time frames. We tried to remedy this issue by applying stratified sampling: the dataset is split in such a way that ensures that the model sees at least some examples from each possible combination of 'time of the day' and 'day of the week' features during training. Moreover, the low number of passengers on many timing points makes it harder to give an accurate prediction. If only one or two people enter at a certain timing point, this might cause huge variance in the data. However, the predictions of the total number of passengers seemed to be accurate enough, so we decided to keep the simple feedforward architecture of the model.

In order to obtain the final prediction of the total crowdedness, we feed the output of the Seq2Seq model into the

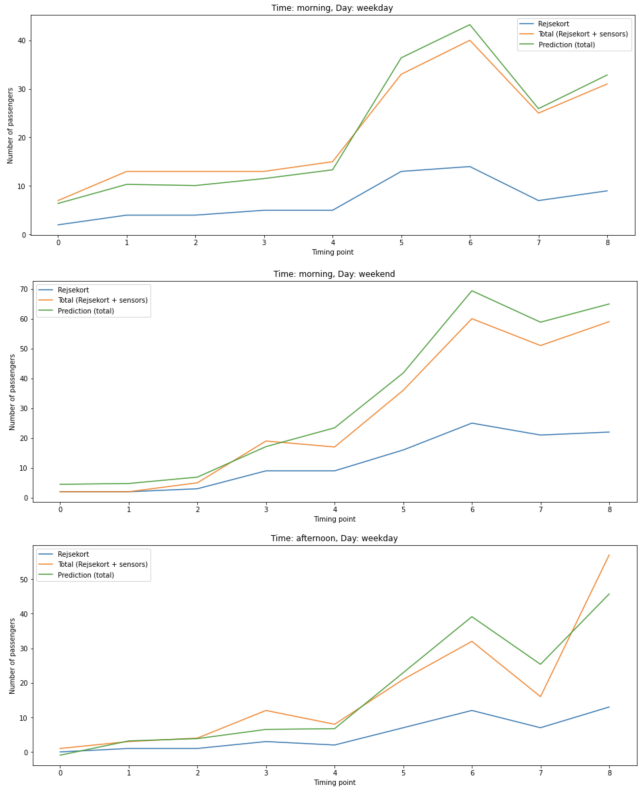

Fig. 7. Examples of predicted total crowdedness based on the Rejsekort data.

feedforward model described above. The result of such an execution is depicted in Fig. 8.

6. CONCLUSION

Although it took us lots of trial and error and multiple approaches to get to good results, in the end we managed to create a neural network that makes good predictions for the crowdedness of the '150S' bus line in Copenhagen. It turned out that the approach with a simple LSTM cell, similar to the one we used in the tutorials, was not suitable to our issue and inadequate for predicting public transport crowdedness. After experimenting with more complex architectures we finally found one based on encoder-decoder architecture that managed to capture demand trends well and generate accurate (Rejsekort) crowdedness predictions.

The prediction of the actual number of passengers based on the number of Rejsekort users, on the other hand, turned out to be an easier problem. Even our initial, quite small and simple, feedforward neural network was enough to give decently accurate results. The biggest problem we faced here was the (relative) scarcity of training data: there were only a few hundred bus rides with sensors on line '150S' in the last several months.

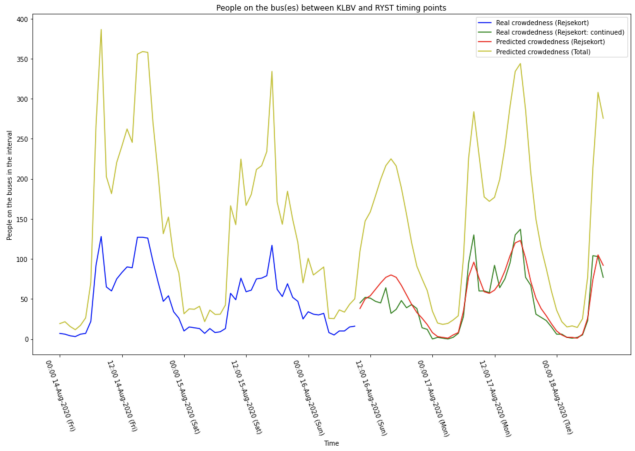

Fig. 8. An example of the final output of both models combined.

As we were limited by the time frame of the project, we were not able to fully explore a range of possible options for improvement of our models. We believe that the most promising area of potential improvements would lie within supplying our main model with spatial features and more external information affecting demand. This means considering not only time of the day or day of the week but also additional external information, such as weather conditions and Danish holidays calendar. Although we briefly looked into incorporating these features into the model, more experimenting would be required to make this work.

We worked on our problem with scalability in mind, so we tried to limit the amount of time needed to train the model. Thus, we believe that our model can be scaled to other bus lines (and directions) in Copenhagen without any significant challenges.

All in all, predicting passenger numbers is not just important for transport companies to make business decisions and keep public transport running. Nowadays, it is also vital in stopping the spread of Covid-19 amidst the 2020-2021 pandemic and possible similar outbreaks in the future. Social distancing is not possible in overcrowded buses and trains. This problem becomes especially pressing in big cities like Copenhagen, where many people rely on public transport. Accurate predictions of crowdedness in public transport can help both passengers and transport companies to make decisions that limit unnecessary contact between people and possible exposure to the virus.

7. REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," 2016.