Chair of Transportation Systems Engineering
Department of Mobility Systems Engineering
Technical University of Munich

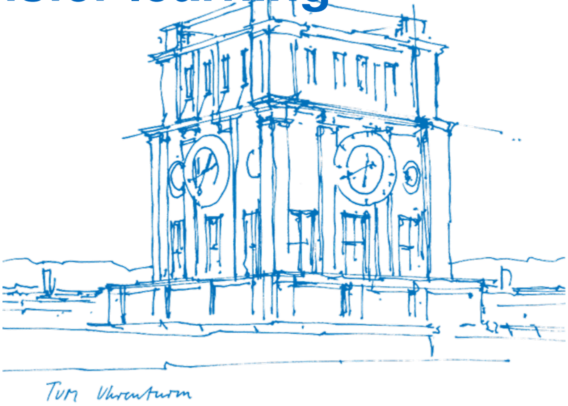
Improving spatio-temporal traffic prediction through transfer learning

Mid-term Presentation

João Rodrigo Olenscki

Chair of Transportation Systems Engineering
Department of Mobility Systems Engineering
Technical University of Munich

December 8th, 2023



Outline

Recap

- Main objective: improve ST traffic prediction via transfer learning.
- Methodology: Typically, the process involves initially training a model using a comprehensive dataset (referred to as the source), followed by conducting a domain adaptation to effectively apply this model to a smaller dataset (identified as the target).
- Task list:

  - ✓ Bibliographic Revision
  - ✓ Model draft
  - ✓ Implementation of the Feature Extractor
  - ··· Evaluation of Feature Extractor's parameters
  - ☐ Implementation of the Predictor
  - ☐ Evaluation of Predictor's parameters
  - ☐ Model revision
  - ☐ Final testing
  - ··· Thesis writing

Problem

- Given an input of 1 hour of traffic data from a city, we want to develop a model that can predict the traffic evolution through the next 1 hour;
- Our dataset comprises 8 cities each with 6 months of 5-minutes spaced snapshots;
- Furthermore, we want to do so in a city with limited amount of data by leveraging knowledge from data rich cities;
- Challenges faced include disbalance on the dataset (most of the data is null), overfitting, problems during domain adaptation.

Outline

Proposed Model Overview

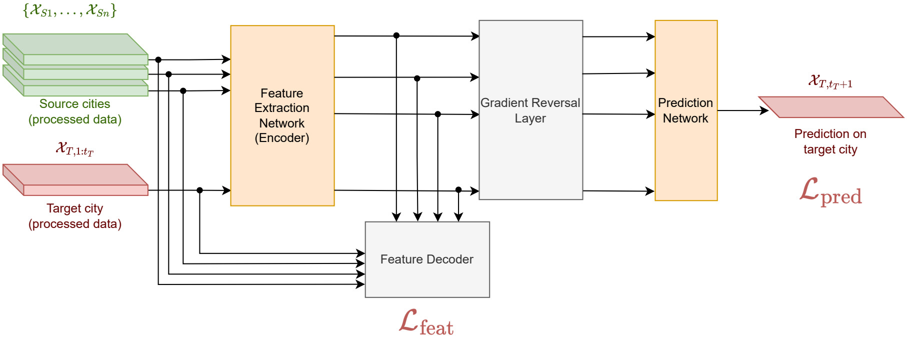


Figure 1 Proposed Model.

Model Components

- Feature Extractor: Responsible for extracting Spatio-Temporal (ST) features from the data; modeled as an autoencoder.
- Domain Adaptor: Facilitates domain adaptation from the source cities to the target city; implemented using a Gradient Reversal Layer.
- Predictor: Makes predictions on the next time steps of the target data, leveraging the knowledge acquired from the source data.

Note that by modeling the Feature Extractor as an autoencoder, it can be trained separately from the rest of the model, enhancing flexibility.

Feature Extractor/Autoencoder

- Architecture:
  - comprises an encoder and a decoder.
  - encoder reduces data dimensionality, capturing latent features.
  - decoder reconstructs data from these features, ensuring feature relevance.

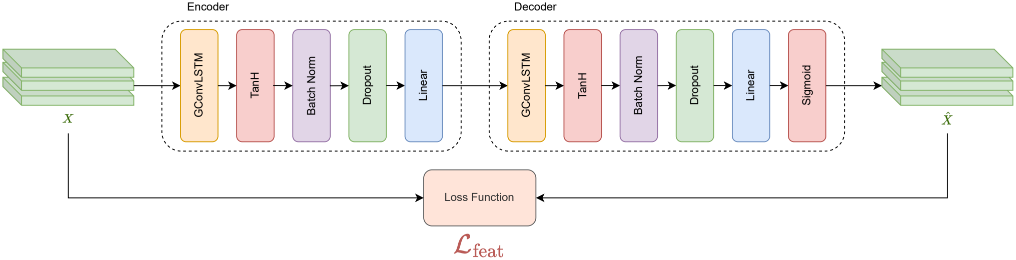


Figure 2 Autoencoder architecture.

Feature Extractor/Autoencoder

- Implementation:
  - the GConvLSTM cell applies convolutions (CNN) to the input graph and uses the convoluted tensor as an input to a LSTM layer. The convolutions aim to capture spatial features, while the LSTM, a recurrent neural network, extracts temporal features.
  - we apply then an activation function (TanH), followed by a batch normalization, a dropout regularization, and a linear layer (with an extra activation for the decoder).
  - the original data and the reconstruct one are then tested against each other using a loss function.

Outline

Overview

- Experiments 1 through 5 were dedicated to parameter exploration, encompassing:
  - Exploring the optimal dimensions for the output of convolutional and linear layers, i.e., the latent space dimension.
  - Investigating the influence of the number of source cities on the autoencoder's performance, excluding domain adaptation considerations.
  - Assessing the impact of the degree of Chebyshev polynomials, a crucial parameter in GConvLSTM, on overall data reconstruction.
  - Examining various activation functions for both the encoder and decoder, to determine their effect on model efficacy.
- Experiment 6 investigates the effectiveness of knowledge transfer in the autoencoder by pretraining it with source data.

Findings

- From experiment 1 (impact of the `K_cheb` parameter):
  - values tested: $1, 2, 3, 4, 5, 6$;
  - small values decrease the performance significantly;
  - larger values have increased computational cost (as a bigger computational graph needs to be created for each backpropagation call);
  - for our setup (reduced city size), too large values cause overfitting.
- From experiment 2 (impact of the number of source cities `N_cities`):
  - values tested: $1, 2, 4, 8$;
  - the larger the number of cities the better the reconstruction;
  - but again, the increase in the size of the dataset increases the training time.
- From experiment 3 (impact of the activation function):
  - functions tested: Sigmoid, ReLU, TanH;
  - didn’t impact much on performance.

Findings

- From experiment 4 (impact of the loss functions):
  - functions tested: MSE, MSLE, LogCoshLoss, CustomHuberLoss, FocalLoss, ZeroInflationLoss;
  - best performance by ZeroInflationLoss, a variation of the MSE which is defined below, with weight defined as $w = 100$ for non-zero values and $w = 1$ for zero values.

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} w \cdot (y_i - \hat{y}_i)^2 \tag{1}$$

Findings

- From experiment 5 (impact of the inner layer dimensions):
  - pairs for `conv_dim` and `latent_dim` tested: $(32, 16), (32, 8), (16, 16), (16, 8), (16, 4), (8, 8), (8, 4), (8, 2)$
  - generally, the greater the values of the latent dimensions, the better the model could generalize the features and reconstruct the original data;
  - best results for $(32, 16)$ and $(16, 4)$;
  - for $\texttt{conv\_dim} = 8$ the results are generally bad.

Note that `conv_dim` refers to the number of output channels of the `GConvLSTM` cell and `latent_dim` is the output of the linear layer.

Findings

- Experiment 6 consisted in:
  1. train the autoencoder with the limited target data only;
  2. create a new autoencoder, train it first with data from one source city;
  3. reset the optimizer, fine-tune by training it on the target data;
  4. compare the results (reconstruction) of both autoencoders.
- Findings:
  - pre-training seems to be a suitable approach for domain adaptation;
  - we could make decrease the testing MSE in $\sim 4\times$.

Outline

Next Steps

- Conduct further experiments to assess the influence of an increased number of source cities on domain adaptation effectiveness (Expanding Experiment 6 with a variable number of source cities).
- Develop and integrate the Predictor block into the model.
- Systematically evaluate the performance of the Predictor block.
- Undertake a comprehensive evaluation of the overall model to validate its effectiveness in traffic prediction.
- Continuously work on the thesis writing in parallel with the experimental and development phases.