Traffic Prediction for Smart City Transformation

Introduction

In the journey toward transforming our city into a smart and intelligent urban environment, the government has recognized the critical importance of efficient traffic management. As part of this initiative, our task is to predict traffic patterns across four key junctions for the upcoming four months. By accurately forecasting traffic trends, we aim to optimize city services, enhance citizen experiences, and lay the groundwork for future infrastructure planning.

Exploratory Data Analysis (EDA)

Our initial dive into the dataset unearthed valuable insights:

- Data Richness: The dataset comprises 48,120 data points, collected hourly over a span of 20 months. This wealth of data provides a comprehensive foundation for analysis and modeling.
- Data Challenges: Some junctions exhibit data sparsity, presenting challenges in capturing accurate traffic patterns. Addressing these gaps requires thoughtful modeling strategies and feature engineering.
- Temporal Dynamics: The temporal coverage from November 1, 2015, to June 30, 2017, reveals nuanced traffic variations across different time periods, including weekdays, weekends, and holidays.

Approach

To tackle the complex task of traffic prediction, we adopted a sophisticated neural network architecture, leveraging modern deep learning techniques. Key components of our approach include:

- Architecture Design: We opted for a multi-attention Recurrent Neural Network (RNN) architecture, chosen for its ability to capture temporal dependencies and complex patterns in sequential data.
- Attention Mechanism: Incorporating attention layers enabled our model to focus on relevant temporal features, such as lagged traffic patterns at various time intervals (e.g., days, weeks, months).

- Temporal Feature Engineering: By encoding historical traffic data spanning different time windows, we aimed to capture the diverse temporal dynamics influencing traffic flow at each junction.
- Teacher Forcing Strategy: Employing teacher forcing during training facilitated the prediction of 48-hour intervals based on actual historical data, enhancing model robustness and accuracy.

Architecture

Our neural network architecture comprises the following components:

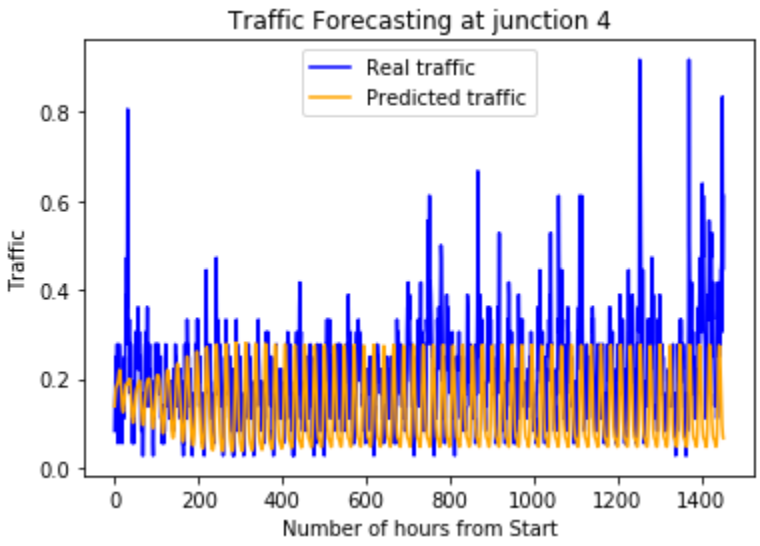
- Encoder RNN: Implemented using CuDNNGRU layers, the encoder captures sequential dependencies in the input data, encoding temporal information into latent representations.
- Attention Decoders: Multiple attention layers are employed to extract relevant temporal features from different time intervals, allowing the model to adaptively weigh the importance of past traffic patterns.
- Decoder RNN: The decoder, also powered by CuDNNGRU layers, processes the concatenated attention features, generating predictions for future traffic patterns.
- Regressor Component: A dense regression layer is utilized to produce traffic predictions, with the model trained to minimize the root mean squared error (RMSE) loss.

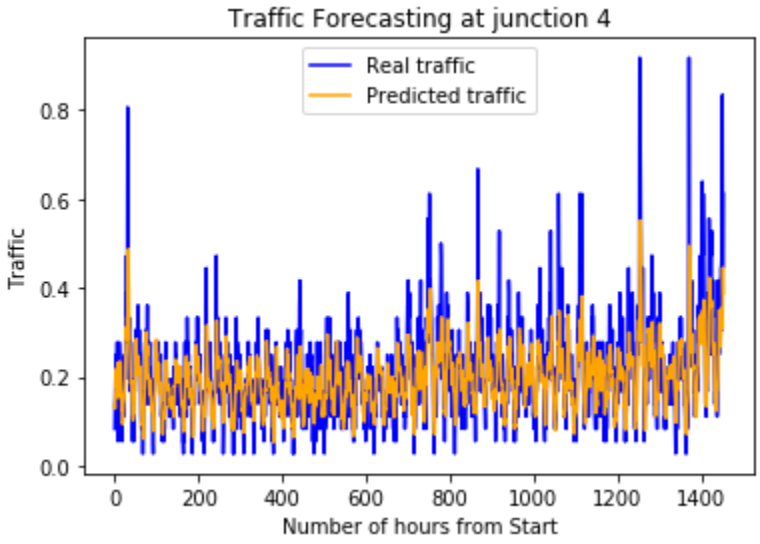
Results

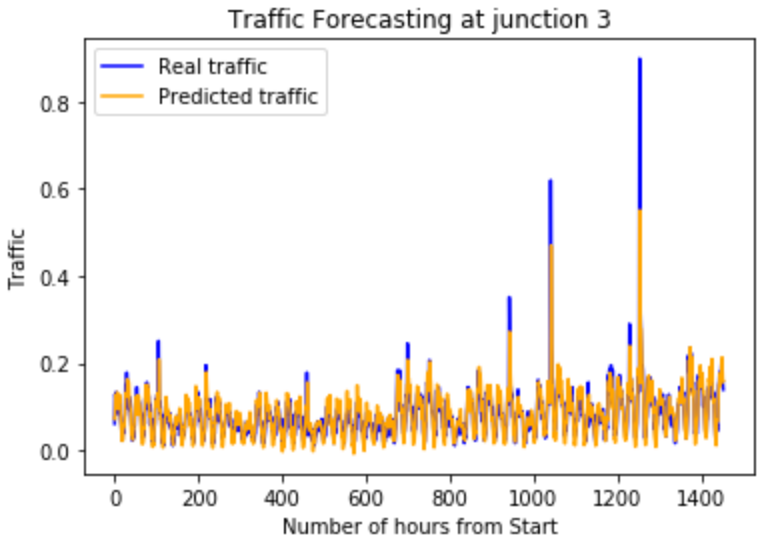
Evaluation of our model yielded promising outcomes:

- Teacher Forcing Evaluation: Predicting 48-hour intervals based on actual historical values demonstrated satisfactory performance, indicating the model's ability to capture short-term traffic fluctuations accurately.
- Long-term Projections: Despite initial challenges, predicting the entire three-month period based on previous predictions showcased the model's potential for capturing evolving traffic trends. However, optimization and fine-tuning are required to address issues such as sinusoidal predictions and sensitivity to optimizer choices.
- Optimization Sensitivity: Our experiments highlighted the significant influence of optimizer selection on model performance, underscoring the importance of rigorous experimentation and hyperparameter tuning.

Implementation







Future Work

Looking ahead, several avenues for future enhancement and exploration have been identified:

- Framework Migration: Consider transitioning the model to dynamic deep learning frameworks like PyTorch to facilitate state sharing between RNNs and enable more efficient training.
- Model Stacking Strategies: Explore ensemble approaches by combining our RNN-based model with classical time-series models (e.g., ARMA, ARIMA) and gradient boosting models (e.g., XGBoost) to leverage their complementary strengths and improve predictive accuracy.
- Optimization Refinement: Conduct thorough optimization and hyperparameter tuning experiments to address model performance issues, including sinusoidal predictions and optimizer sensitivity, thereby enhancing the model's robustness and generalization capabilities.