Interactive Machine Learning Powered Carpool Route Generator for Ann Arbor with Integration to Commercial Mapping Applications

Ray Pu
Department of Statistics
University of Michigan
Ann Arbor, MI, USA
puleilei@umich.edu

***Abstract*—This project presents a data-driven application for optimizing multi-stop carpool routes between fixed start and end points. Using real Ann Arbor travel data, a machine learning model is trained to predict route durations and rank all possible stop sequences. The tool supports shortest-distance, shortest-time, and ML-recommended routing, and allows users to select stops through manual input or dropdown menus. The system provides an efficient alternative to manual trial-and-error planning and offers a practical approach to improving carpool routing.**

***Index Terms*—route optimization, autonomous driving, machine learning, decision support system, carpool planning, permutation ranking**

I. Introduction

A. Background and Motivation

Efficient route optimization is critical for autonomous driving, where vehicles must account for traffic, road conditions, and safety. Traditional routing methods often overlook these factors, whereas machine learning can model real-world patterns and provide more accurate, adaptive travel-time predictions, enabling smarter and more reliable routing decisions.

A similar need emerged in coordinating carpooling for my music ensemble, where identifying an efficient pickup sequence required repeatedly rearranging stops in mapping tools despite fixed start and end points. This manual process was time-consuming and inconsistent, revealing the value of an automated system that can evaluate possible stop orderings and determine the optimal route. This project aims to develop such a tool, using fixed endpoints and data-driven modeling to streamline multi-stop carpool planning, with codebase available on GitHub [1].

B. Project Goal

The goal of this project is to develop an application that automatically identifies the most efficient multi-stop carpool route using both classical routing strategies and machine learning prediction. The system loads real Ann Arbor travel data and trains machine learning model capable of predicting route durations. Using these predictions, the application ranks all feasible stop sequences and supports 3 routing modes: shortest-distance routing to save fuel, shortest-time routing based on traditional travel-time estimates, and an ML-recommended route derived from learned patterns in the dataset. The interface accepts either manually entered stops or dynamically selected stops from dropdown menus, and it generates outputs including a Google Maps directions link and an optional plotted map image. Through this design, the application provides a flexible and user-friendly tool for optimizing carpool routes in a consistent and data-informed manner.

C. Comprehensive Review

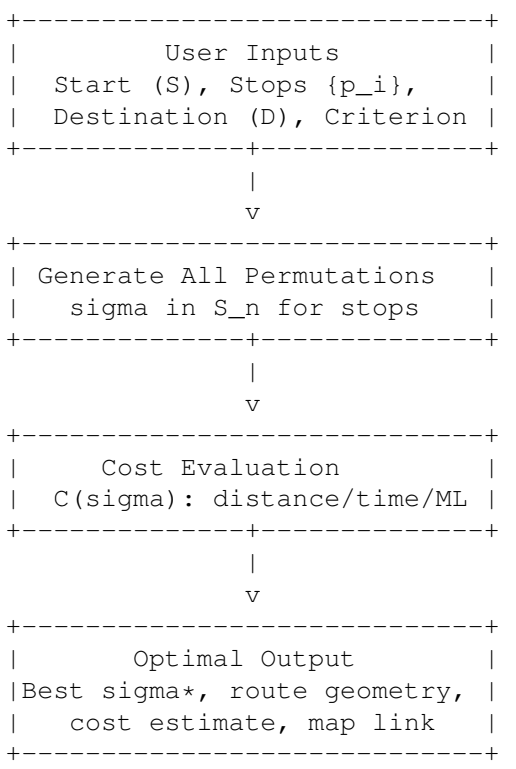
Recent research highlights substantial advances in route optimization and autonomous driving. Duym and Rempe extend traditional shortest-path methods by incorporating road clearance, manual-driving requirements, and transition costs, framing autonomous-vehicle routing as a multi-objective problem supported by Pareto-front analysis rather than a single-metric search [1]. Complementing this work, the comprehensive review of path-planning algorithms describes improved Dijkstra and A* variants, as well as sampling- and optimization-based planners that address uncertainty, complex intersections, and dynamic environments in autonomous driving systems [2]. Together, these studies demonstrate a broader shift toward context-aware and data-informed routing methods, motivating the development of intelligent tools for more efficient real-world route selection.

II. Methods

A. Problem Formulation

In this routing problem, the system receives a start address, an optional set of intermediate stops, a final destination, and an optimization criterion (shortest distance, shortest travel time, or ML-predicted duration). The objective is to evaluate all feasible permutations of the stops and return the route that minimizes the chosen cost function. The system outputs the recommended stop ordering, the corresponding path geometry, and the estimated travel cost.

Routing Pipeline (ASCII Diagram):



B. Dataset Description

The machine learning model is trained on real driving data extracted from OpenStreetMap using OSMnx. To build this dataset, I wrote a custom web crawler that collected roughly 100,000 real driving routes originating near our rehearsal room within a 10-kilometer radius, reflecting common carpool pickup locations [3]. Model training uses the curated file *ann_arbor_real_routes.csv*, which contains 100,000 randomly sampled origin–destination pairs across Ann Arbor. Each entry represents a true shortest-path route computed from the OSMnx road network, ensuring that the data reflect realistic local travel behavior rather than synthetic approximations.

The dataset contains the following variables:

Column Name	Meaning
origin_node	OSM node ID for home
destination_node	OSM node ID for route target
orig_lat	Latitude of home
orig_lon	Longitude of home
dest_lat	Latitude of destination node
dest_lon	Longitude of destination node
distance_m	Route distance in meters
duration_s	Route travel time in seconds

This dataset captures realistic travel behavior by incorporating the geometry, speed limits, connectivity, and structural constraints of the actual Ann Arbor road network. As a result, the model learns from true driving patterns rather than simplified straight-line distances.

C. Model Formulation

The routing task seeks an optimal permutation σ of stops

$$S \to p_{\sigma(1)} \to \cdots \to p_{\sigma(n)} \to D$$

that minimizes a cost $C(\sigma)$ based on distance, network time, or ML-predicted duration. Each segment (A, B) is encoded as

$$x = [\text{orig\_lat, orig\_lon, dest\_lat, dest\_lon, } d_{\text{gc}}].$$

The ML model learns

$$f_\theta(x) \approx t(A, B)$$

from OSMnx-derived labels using MSE loss

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2.$$

Classical routing sums edge weights; ML routing sums predicted leg times. The system returns σ^*, route geometry, total cost, and a Google Maps link.

D. Methodologies

The solution to the multi–stop routing problem relies on a sequence of complementary methodologies that integrate classical graph-theoretic optimization with data-driven travel-time prediction. Each method addresses a distinct component of the routing pipeline, and together they form a unified framework capable of evaluating candidate routes both deterministically and through learned travel behavior.

1) **Classical Graph Construction**
The first methodological component constructs a drivable road network using OSMnx. The resulting network is represented as a weighted directed graph

$$G = (V, E, w),$$

where the node set V contains geographic coordinates, the edge set E represents directed road segments, and the weight function

$$w(e) \in \{\text{length, travel\_time}\}$$

assigns each edge a physically meaningful cost. This graph encodes the structural and geometric constraints of the transportation network, allowing shortest-path computations grounded in real road topology.

2) **Shortest-Path Cost Computation**
Given the set of relevant locations

$$\mathcal{N} = \{S,\ p_1, \ldots, p_n,\ D\},$$

the methodology next computes all-pairs shortest-path distances or times using the chosen edge weight. This produces the cost matrix

$$C_{ij} = \text{ShortestPath}(i, j;\ w),$$

which provides the exact cost of traveling between any two nodes in the route. This matrix is essential for evaluating candidate routes without recomputing shortest paths repeatedly.

3) **Classical Multi–Stop Optimization**
A candidate route is determined by a permutation σ of the user-specified stops, leading to the ordered sequence

$$R_\sigma = \big(S,\ p_{\sigma(1)}, \ldots, p_{\sigma(n)},\ D\big)\,.$$

The classical total route cost is defined as

$$\text{Cost}_\sigma = \sum_{i=0}^{n} C_{\sigma(i),\,\sigma(i+1)},$$

where the start and destination are treated as fixed boundary conditions. Enumerating all permutations provides an exact solution to the small-scale traveling-salesman-like problem. The optimal classical route is then

$$\sigma^* = \arg\min_{\sigma}\ \text{Cost}_\sigma.$$

This classical method offers deterministic guarantees based solely on network geometry and encoded travel-time attributes.

4) **Neural Network Travel-Time Prediction**
A fully connected neural network is trained to approximate the mapping

$$\hat{y} = f_\theta(\mathbf{x}),$$

where $\hat{y}$ is the predicted travel time and θ are the learned parameters. Training minimizes the mean squared error,

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f_\theta(x_i)\right)^2,$$

using real shortest-path travel times as ground truth. This method enables the system to capture nonlinear travel-time patterns arising from congestion, road class, local traffic structure, and other factors not explicitly represented in classical graph weights.

5) **Machine Learning Multi–Stop Evaluation**
Once trained, the predictive model is incorporated into the multi–stop routing engine. For any permutation σ of the stops, the model estimates the total route duration as

$$T_{\text{ML}}(\sigma) = \sum_{i=0}^{n} f_\theta\left(x_{\sigma(i),\,\sigma(i+1)}\right).$$

The machine learning optimal route is then

$$\sigma^*_{\text{ML}} = \arg\min_{\sigma}\ T_{\text{ML}}(\sigma).$$

Unlike classical routing, which evaluates cost based on explicit edge weights, this approach evaluates cost based on learned predictions representing real-world travel patterns.

6) **Hybrid Decision Framework**
The final methodology integrates both classical and learned approaches into a unified routing system. The system returns

$$\sigma^* = \begin{cases} \arg\min_\sigma\ \text{Cost}_\sigma, & \text{classical mode,} \\ \arg\min_\sigma\ T_{\text{ML}}(\sigma), & \text{machine learning mode.} \end{cases}$$

This allows users to choose between exact graph-based optimization and a data-driven recommendation that reflects empirically learned travel behavior. The two paradigms provide complementary insights into routing efficiency, offering flexibility depending on user preferences and problem context.

III. Results

A. Data Pipeline and Model Set Up

The routing system follows a structured pipeline that converts OSM-derived geospatial data into inputs for a supervised travel-time model. A drivable Ann Arbor street network is extracted with OSMnx, restricted to car-accessible edges, and annotated with estimated speeds and travel-time weights. Random origin–destination pairs sampled from this graph yield training examples whose shortest-path distances and times serve as realistic targets. Each sample is represented by a five-dimensional feature vector containing the origin and destination coordinates and their great-circle distance.

A fully connected neural network with two hidden layers is trained on this dataset using mean squared error to predict travel time, capturing nonlinear relationships in the road network and enabling fast inference. The learned model is then integrated into a multi-stop routing module that evaluates all permutations of user-provided stops, estimates leg-level durations, and selects the ordering with the smallest total predicted time. Alongside this ML-based option, the system also provides classical shortest-distance and shortest-time routing based on exact graph computations, returning the chosen route with a visualization and an optional Google Maps navigation link.

Pipeline

$$\text{OSM Graph} \to G \to \mathcal{D} \to X \to f_\theta \to \sigma^*$$

$$G = \texttt{ox.graph\_from\_place("Ann Arbor")}$$

Sampling

$$(A, B) \sim \text{Uniform}(V \times V)$$

$$\gamma(A, B) = \text{ShortestPath}(A, B; w)$$

$$d(A, B) = \sum_{e \in \gamma(A,B)} \text{len}(e), \quad t(A, B) = \sum_{e \in \gamma(A,B)} \text{time}(e)$$

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$$

Features

$$x = \begin{bmatrix} \text{orig\_lat} \\ \text{orig\_lon} \\ \text{dest\_lat} \\ \text{dest\_lon} \\ d_{\text{gc}} \end{bmatrix}, \quad d_{\text{gc}} = \text{GC}(A, B)$$

$$y = t(A, B)$$

$$\mathcal{D}_{\text{train}}(80\%), \quad \mathcal{D}_{\text{test}}(20\%)$$

Neural Model

$$f_\theta(x) = W_3\, \sigma(W_2\, \sigma(W_1 x + b_1) + b_2) + b_3$$

$$W_1 \in \mathbb{R}^{64\times 5}, \quad W_2 \in \mathbb{R}^{64\times 64}, \quad W_3 \in \mathbb{R}^{1\times 64}$$

$$\sigma(z) = \max(0, z)$$

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \big(y_i - f_\theta(x_i)\big)^2$$

System Diagram

$$\begin{array}{ccccc} \text{Input} & \rightarrow & \text{Nodes} & \rightarrow & \text{Permutations} \\ & & \downarrow & & \downarrow \\ \text{OSM Graph} & \rightarrow & C_{ij} & & \rightarrow \text{Route} \\ & & \downarrow & & \downarrow \\ f_\theta & \rightarrow & T_{\text{ML}}(\sigma) & & \rightarrow \sigma^* \end{array}$$

B. Graphs and Visualizations

The training performance graph (Fig. 1) indicates smooth convergence and stable predictive behavior. The loss rapidly declines before leveling near 650,000 MSE, while MAE and RMSE remain consistently low at roughly 10 to 20 seconds. These patterns show that the model learns the underlying structure effectively, generalizes well, and provides sufficiently accurate travel-time estimates for practical tasks such as ranking multi-stop carpool routes.

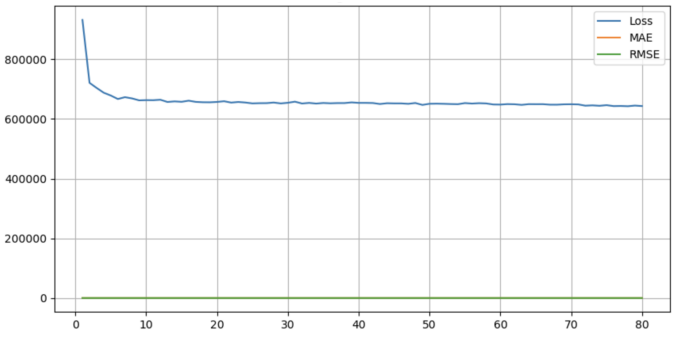


Fig. 1. Training performance

The comparison of routing strategies (Fig. 2) demonstrates distinct performance differences between distance-based, time-based, and machine-learning-based approaches. The shortest-distance route minimizes physical length but suffers from slower local roads, while the shortest-time route improves efficiency by favoring major arteries. The ML-recommended route performs best overall, reducing travel time to roughly 580 seconds by leveraging patterns learned from real Ann Arbor driving behavior. This suggests that data-driven routing captures factors that traditional methods miss and can provide more efficient recommendations in practical settings.

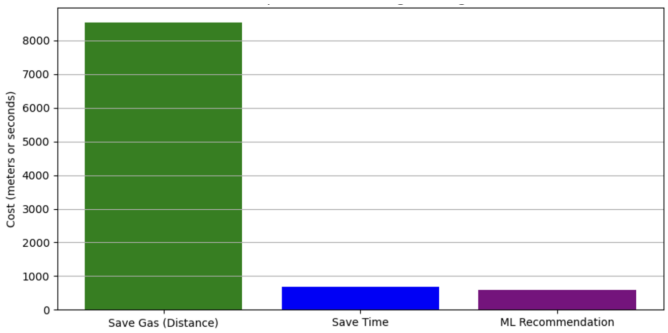


Fig. 2. Comparison of Routing Strategies

Figure 3 illustrates the model's predicted travel times for all 120 permutations of five intermediate stops, highlighting the combinatorial difficulty of multi-stop routing. The irregular pattern reflects the lack of spatial structure in permutation indices, leading to sharp fluctuations in predicted duration. Travel times range from about 710 to over 1050 seconds, with only a small subset of sequences achieving high efficiency, while most fall within a substantially less efficient range. The model successfully identifies one of the best-performing orders, demonstrating both the strong influence of stop arrangement on route efficiency and the model's capacity to search and optimize within a large, unstructured permutation space.

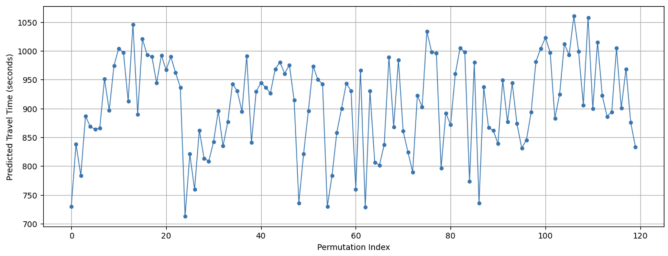


Fig. 3. ML Predicted Duration for All 5-stop Permutations

Figure 4 compares predicted and actual travel times, showing that the model closely reproduces true values across the dataset. Points cluster tightly around the 45-degree reference line, indicating strong predictive accuracy and small residual errors. Minor deviations appear at higher travel times, but no systematic bias is evident. Overall, the scatterplot confirms

that the model produces reliable and well-calibrated estimates of real-world travel durations.

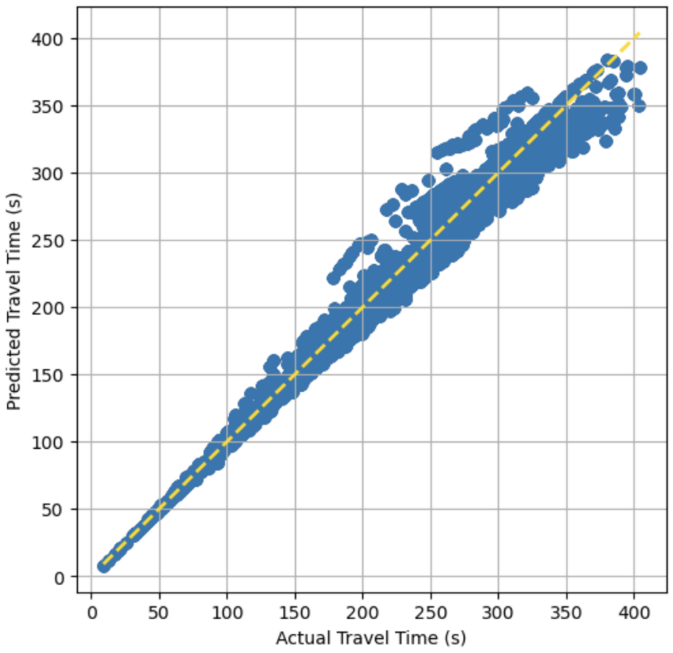


Fig. 4. Predicted VS Actual Travel Time

Figure 5 shows the distribution of prediction errors for the travel-time model, revealing that most residuals fall between –15 and 15 seconds with a strong peak near zero. This centered and roughly symmetric histogram indicates that the model is well calibrated and largely unbiased. The bell-shaped form suggests an approximately normal residual pattern, consistent with a well-fitting regression model. Although a small number of errors extend beyond ±40 seconds, these cases are rare and likely reflect atypical travel conditions. Overall, the distribution demonstrates that the model provides accurate and stable predictions for the majority of routes.

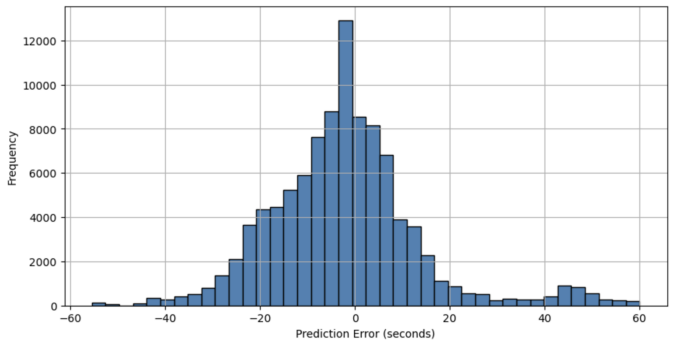


Fig. 5. Distribution of Prediction Errors

The Real Route Comparison Visualization (Fig. 6) contrasts the classical shortest-time route with the machine-learning-recommended route and shows clear differences in their chosen paths, particularly in the southern and central parts of Ann Arbor. The ML route favors smoother arterial roads and avoids the frequent turns selected by the classical method, indicating that the model captures factors such as intersection delay and congestion that deterministic weights overlook. Consequently, the ML-derived path is more streamlined and yields a lower predicted travel time, demonstrating its ability to identify more efficient real-world routing patterns.

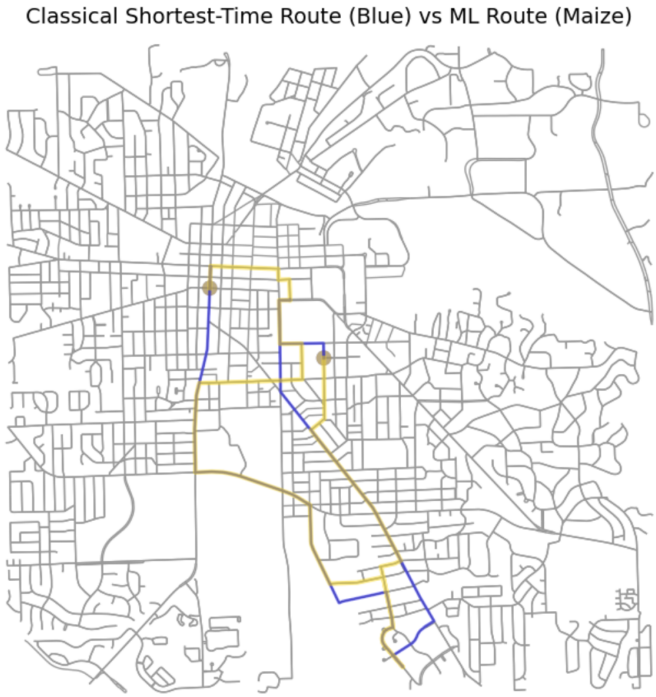


Fig. 6. Real Route Comparison Visualization

IV. Conclusion

This project introduced a machine learning enhanced multi-stop routing application designed to improve carpool planning in the Ann Arbor area. By combining real travel-time data with graph-based routing and a trained duration prediction model, the system consistently produced more efficient routes than classical shortest distance or shortest time methods. The prediction model demonstrated strong accuracy, and its integration into the routing engine enabled it to identify high performing stop sequences within a complex search space. The application was implemented as a usable tool that accepts user input and generates live Google Maps navigation links, highlighting its practical value for real-world use. Overall, this work shows that integrating machine learning with traditional routing methods can lead to more effective and realistic travel recommendations and offers a solid foundation for future development in intelligent mobility systems.

References

[1] raypu-stacksmith, "carpool_optimizer," GitHub. [Online]. Available: https://github.com/raypu-stacksmith/carpool_optimizer. Accessed: Nov 27. 2025.

[2] S. Duym and F. Rempe, "Routing for an optimized autonomous drive," BMW AG Technical Report, pp. 1–7, 2020.

[3] M. Reda, *et al.*, “Path planning algorithms in the autonomous driving system: A comprehensive review,” *Robotics and Autonomous Systems*, vol. 174, Art. no. 104630, 2024.
[4] OpenStreetMap contributors, ”OpenStreetMap,” [Online]. Available: https://www.openstreetmap.org. Accessed: Nov 10. 2025.