RouteLLM: A Large Language Model with Native Route Context Understanding to Enable Context-Aware Reasoning

PHILIPP HALLGARTEN, Porsche AG, Germany and Technical University of Munich, Germany
VERENA JASMIN HALLITSCHKE, Porsche AG, Germany
ENKELEJDA KASNECI, Technical University of Munich, Germany
MICHAEL BEIGL, Karlsruhe Institute of Technology, Germany
TOBIAS GROSSE-PUPPENDAHL, Porsche AG, Germany

Understanding users' environments is crucial for determining their states, needs, and interactions with technology. This work focuses on route context, including environmental factors such as road conditions, traffic, and weather that influence users while traveling. Integrating route context with LLMs enables reasoning over environmental factors, thus allowing users to ask questions like 'When is the best moment for a phone call along my route?' or 'Is this a good route for a drive in a convertible?'. We introduce the first LLM that natively understands route context. We create *ContextualRoutes*[1], a dataset of 320k routes, each comprising road, weather, and traffic data. We annotate these routes using a template and a teacher model to create *LabeledRoutes*[1], a multimodal multi-task question-answering dataset with over 1k tasks and 40k conversations containing routes and text. Based on the first dataset, we train the first route context tokenizer that groups the routes into semantically meaningful clusters. On its basis, we propose the first route-context-aware LLM and find it capable of zero-shot reasoning on routes. Still, we urge that further research on learning cross-modal route-to-text understanding is necessary and discuss challenges in the future development of artifacts for this novel branch of research.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computing methodologies** → *Knowledge representation and reasoning*.

Additional Key Words and Phrases: Route Context; Context-Aware LLMs

ACM Reference Format:
Philipp Hallgarten, Verena Jasmin Hallitschke, Enkelejda Kasneci, Michael Beigl, and Tobias Grosse-Puppendahl. 2025. *RouteLLM*: A Large Language Model with Native Route Context Understanding to Enable Context-Aware Reasoning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 9, 3, Article 83 (September 2025), 34 pages. https://doi.org/10.1145/3749552

1 INTRODUCTION

Since the release of ChatGPT[2], the rising popularity of Large Language Models (LLMs) has revolutionized how we interact with technology. Due to LLMs showing promising reasoning capabilities and factual knowledge, there have been intensive efforts to include LLM-based interfaces in a growing number of applications, for example,

[1]we made the *ContextualRoutes* dataset available under Open Database License (ODbL), and the *LabeledRoutes* dataset under the Open Data Commons Attribution License (ODC-By), at https://huggingface.co/RouteLLM-Dataset
[2]https://openai.com/chatgpt/, last accessed July 16, 2025

Authors' Contact Information: Philipp Hallgarten, Porsche AG, Stuttgart, Germany and Technical University of Munich, Munich, Germany, philipp@phallgarten.com; Verena Jasmin Hallitschke, Porsche AG, Stuttgart, Germany, verenahallitschke@gmail.com; Enkelejda Kasneci, Human-Centered Technologies for Learning, Technical University of Munich, Munich, Germany, enkelejda.kasneci@tum.de; Michael Beigl, Karlsruhe Institute of Technology, Karlsruhe, Germany, michael.beigl@kit.edu; Tobias Grosse-Puppendahl, Porsche AG, Stuttgart, Germany, tobias@grosse-puppendahl.com.

ACM 2474-9567/2025/9-ART83
https://doi.org/10.1145/3749552

smart assistants on mobile devices [5]. Such systems allow for interactions through a natural language interface, making them simple and intuitive for their users. However, a critical limitation of such LLM-based systems is that they can only process text, images, or audio, thus requiring the encoding of all inputs as one of these modalities. While current LLMs support very large context lengths of up to one million tokens, representing multivariate route context such as weather, traffic, and road states purely in text or image form remains inefficient and indirect. For instance, to describe the dynamic state of a road segment, e.g., its curvature, slope, surface type, or traffic flow, many text tokens or pixels values are required. Additionally, each of these tokens then needs to be encoded and interpreted by the model to infer its semantic meaning. This overhead becomes especially pronounced when dealing with longer sequences of route context, where thousands of segments must be represented. Consequently, modern LLMs are not well-suited to understand the user's environment in this regard, preventing the easy creation of context-aware, empathic, and user-centric systems on the basis of them.

In this work, we take a first step towards solving this problem by focusing on the specific set of environmental contexts that describes the route a user is traveling along. We envision an LLM that can natively understand and reason with route context data, thus allowing users to ask questions that require reasoning about the route they are traveling along, for example, *"What is the best time for a phone call along my route"?* or *"Is this a scenic route?"*.

To lay the foundation for this long-term goal, we create two large-scale route datasets and make them publicly available. The *ContextualRoutes* dataset comprises approximately 325k routes from 18 locations with a total length of over 8.9 million kilometers. In total, the dataset comprises 38 route context features describing road properties (e.g., the number of lanes), the weather, and the traffic along each segment of the route. Based on this dataset, we create the *LabeledRoutes* dataset, a multimodal multi-task dataset that allows to instruction-tune LLMs on routes, containing over 1 000 tasks from 36 categories that require feature extraction, reasoning, and understanding of route structure.

We use these datasets to train an architecture similar to *MotionGPT* [28], thus creating the first LLM with token-level route context understanding, referred to as *native* route context understanding in this manuscript. Through a vector quantized-variational autoencoder (VQ-VAE), the approach efficiently converts a series of route context features into a series of route tokens comparable to text tokens in natural language processing (NLP). We demonstrate that these route tokens are semantically meaningful and encode discriminative variances. Next, we train an LLM (Mistral Instruct 7B[3] [27]) in a three-step pipeline to first understand route tokens, then learn a cross-modal understanding between natural language and routes, and finally align the outputs for a specific task. We demonstrate that our final system provides a natural language interface that allows users to naturally interact with routes. In an exploratory analysis, the system shows zero-shot reasoning capabilities on routes, i.e., it is able of answering questions about unseen routes without additional task-specific training.

In summary, this work has four contributions:

- **C1**: Two novel route context datasets *ContextualRoutes* and *LabeledRoutes*
- **C2**: The first Route-Tokenizer that translates segments of route context into tokens that can be processed by downstream LLMs analogously to tokenizers for natural language,
- **C3**: A first prototype for an LLM with token-level route-context understanding that allows users to leverage the reasoning capabilities of modern LLMs for planning and gathering information about their route, and
- **C4**: A detailed discussion on challenges for the future development of LLMs with token-level route-context understanding

2 RELATED WORK

This section outlines the importance of route context, the current state of research on multimodal LLMs, and previous works on geospatially-aware LLMs.

[3] https://mistral.ai/news/announcing-mistral-7b/, last accessed July 16, 2025

2.1 Significance of Context for Ubiquitous Systems

Prior research has demonstrated that context signals, such as location, weather, traffic, road properties, and daytime, strongly influence humans' internal states [9, 10, 36, 44, 47]. This relationship has been leveraged by various previous works to create novel user experiences through context-aware systems. For example, Frison et al. discuss user interfaces for autonomous vehicles that automatically adapt based on traffic scenarios [20], Tavakoli et al. use information on road conditions and users' environments to create hierarchical driver state models [48], and Acer et al. present a conversational agent that provides information about landmarks based on users' environmental sensor data [2]. Further, Sheshadri and Hara propose a context-aware conversational agent [45] that enables conversational localization, i.e., it helps users with indoor localization by asking them questions about their surroundings and matching the information with floor maps. Wut et al. [53] propose a system that predicts the interruptibility of drivers based on context information captured through video cameras and vehicle telemetry. Kari et al. introduced *SoundsRide* [29], a music augmentation system that temporally aligns significant events in the audio signal, e.g., beat drops, to significant changes in the user's environment, e.g., entering a tunnel. Belz et al. transfer the concept of synchronizing auditive affordances with users' context from music to audio books [7]. With *Story-Driven*, they propose a system that generates an audiobook with the plot taking place at landmarks along the route a user is traveling and then synchronizes the time the landmark is mentioned in the story with the time the user passes along.

While such context-aware systems underline the utility, versatility, and thus the importance of understanding context, they all have one key limitation, they are tailored to specific scenarios and may not easily generalize to others. Simultaneously, we observed how research in various areas has experienced a major push through the introduction of LLMs, allowing researchers to rapidly prototype systems with reasoning capabilities that generalize to a variety of problems. By introducing *RouteLLM*, we bring these benefits of LLMs to the research of context-aware systems. Through natively understanding route context, *RouteLLM* can serve as a general system with broad applicability for scenarios that require reasoning and zero-shot capabilities based on context, thus creating more personalized and adaptive experiences.

2.2 Multimodal Large Language Models

Multimodal Large Language Models (MLLMs) are LLMs that combine understanding of text with at least one other modality, thus enabling the user to interact with the modalities in natural language and leverage the reasoning capabilities of LLMs. Previous research explored images [3, 33, 34], videos [56], or audio recordings [18] as additional modality. One approach to creating an MLLM is multimodal instruction tuning [54]. The model is trained using a special instruction tuning dataset that combines both modalities by concatenating embeddings of the second modality to the prompt token embeddings. The method for extracting the embeddings of the second modality differs between models. In [34], the authors use CLIP [39] to extract visual features from images. A linear layer projects the visual features to create the image embeddings. In their later work, the authors replace the linear layer with an MLP to improve the resulting embeddings [33]. The authors create a multi-task visual instruction tuning dataset using image captions and GPT-4 to label their questions. They use a subset of this dataset to align the features by freezing the LLM and CLIP and only training the projection unit. The authors update the projection unit and the LLM during the instruction tuning. Their final models, LLaVA and LLaVA-1.5, achieve state-of-the-art performance on visual benchmarks. The authors of [28] propose MotionGPT, a model that uses a Vector Quantized-Variational Autoencoder (VQ-VAE) as a motion tokenizer to encode human motion sequences into tokens. Trained on the HumanML3D [21] and KIT-ML [38] datasets, MotionGPT handles tasks like motion synthesis and motion captioning by combining motion and language data through a pre-trained FLAN-T5-Base LLM. The model undergoes three training phases: motion tokenization, cross-modal pretraining on

motion-language pairs, and instruction tuning for multi-task motion question answering. MotionGPT outperforms state-of-the-art models in both motion captioning and motion synthesis.

These works successfully extend LLMs capabilities to visual and motion data. While they lack the ability to process complex environmental contexts like route context, we build upon the introduced concepts by introducing a *Route Tokenizer* and a specialized dataset for route context understanding. Unlike previous multimodal models, which primarily focus on visual or motion data, our system targets route-based reasoning and decision-making, to enable a more context-aware and practical application of LLMs.

2.3 Geospatially-aware LLMs

Many automotive systems require geospatial awareness, meaning the ability to understand and interpret geographical locations and spatial relationships between objects. Some studies have demonstrated that LLMs pretrained on natural language corpora, such as GPT-4, exhibit a limited degree of geospatial understanding [11, 37, 41]. For instance, [37] shows that GPT-4 can achieve a B+ on a geographic information systems (GIS) exam. However, processing route context demands even deeper geospatial comprehension, as it involves reasoning based on spatial features along a route. While there are efforts to enhance the geospatial capabilities of LLMs through fine-tuning [32] and prompt engineering [35], none of these works directly integrates geospatial awareness with route context reasoning in LLMs.

In contrast to these studies, our work addresses this gap by introducing a system that natively integrates route-specific contextual features, directly integrating structured route-specific features at the token level. This enables applications where real-time decision-making along a route depends on environmental factors like traffic, weather, and road features. Thus, this work introduces the first framework that combines the reasoning abilities of modern LLMs with route context understanding.

3 METHOD

3.1 Overview

This work aims to develop an LLM that can natively understand route context alongside natural language, enabling novel interactions such as route-based question-answering or making decisions based on environmental factors while traveling, e.g., determining the safest or most convenient time for a phone call along a route. There are several approaches to integrating route context with LLMs. One could encode route data as text or images or even extract visual features using models like CLIP. However, these methods struggle with the dynamic and multi-dimensional nature of route data. Encoding routes as text may oversimplify complex environmental factors, while images fail to capture temporal relationships between consecutive route segments. Additionally, both approaches rely on indirect encoding of route context, requiring verbose textual or visual descriptions, which increases input size and makes processing less efficient. To address this, we propose a more scalable and structured approach by encoding route context into tokens. Inspired by the architecture of *MotionGPT* [28], which processes human motion data with LLMs, we first train a VQ-VAE-based route tokenizer. This tokenizer converts route context vectors into discrete tokens that can be fed into the LLM, similar to how natural language tokens are processed. By using tokens, we ensure that route data is represented in a compact, efficient format that integrates seamlessly with the LLM's existing natural language capabilities. This approach also allows to maintain a prompt size within the context window limits of standard LLM architectures. In the following sections, we describe the architecture and training strategy for *RouteLLM* in detail. We present an overview of *RouteLLM's* system design in Figure 1.

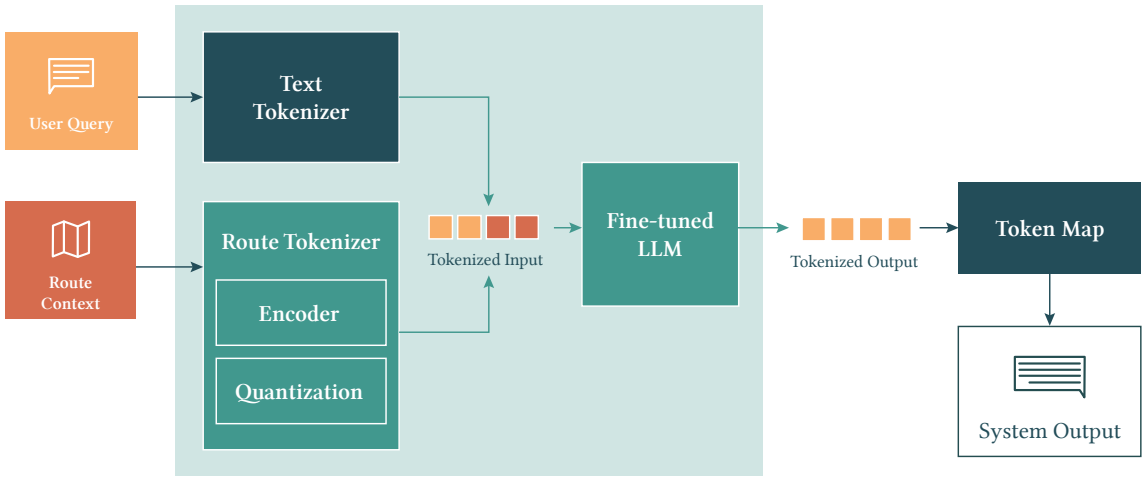


Fig. 1. Overall System Design of *RouteLLM*. We add a VQ-VAE-based Route Tokenizer to a pretrained LLM and fine-tune it for cross-modal understanding between route context and natural language. *RouteLLM* allows users to interact with their route in natural language, and enables tasks that require reasoning and semantic understanding of route context, e.g., finding the best segment for a phone call along a route.

3.2 Definitions

We use a road graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with nodes $\mathcal{N}$ and edges $\mathcal{E}$, where each edge represents a road segment and is described through a set of m road-context features, such as the number of lanes or the class of the road (e.g. a bridge). It is important to note that features can be heterogeneous. We now define a *route* as the sequence of edges connecting two nodes from $\mathcal{N}$, and *route context* as the sequence of corresponding context vectors $[r_i]_{i=1}^{L_R}$ with $r_i \in \mathbb{R}^m$. The goal of this work is to train an LLM so that it understands such route context besides natural language. Hereby, we consider an LLM as a system that processes a sequence of id values, named tokens, $[t_i]_{i=1}^{L_T} t_i \in \mathcal{V}$, and outputs another sequence of tokens from the same set $\mathcal{V}$. Each token usually represents a word piece of natural language and the set of tokens $\mathcal{V}$ is usually referred to as vocabulary. In the LLM, each token is first mapped to an embedding by multiplying a one-hot encoded version of it with a matrix called embedding table, and then processed with a stack of self-attention layers. After the last layer, the outputs are mapped back to a sequence of probability distributions over the vocabulary. To adapt an LLM so that it can process route context besides natural language, we add a second tokenizer to the architecture, dedicated to encoding route context. This *Route Tokenizer* maps a sequence of route context $[r_i]_{i=1}^{L_R}$ to a sequence of route tokens $[u_i]_{i=1}^{L_R}$ with $u_i \in \mathcal{V}^*, \mathcal{V}^* \cap \mathcal{V} = \emptyset$, a corresponding decoder reconstruct the original input from route tokens output by the LLM (see Figure 2). This allows for route context to be used as input to or output by the LLM besides the text tokens. The vocabulary of the LLM is thus updated to $\mathcal{V}' = \mathcal{V}^* \cup \mathcal{V}$, with $\mathcal{V}'$ comprising text and route tokens. We use a specific training strategy to train an LLM that was previously pretrained on a corpus of natural language to learn a cross-modal understanding from text to route context and vice versa.

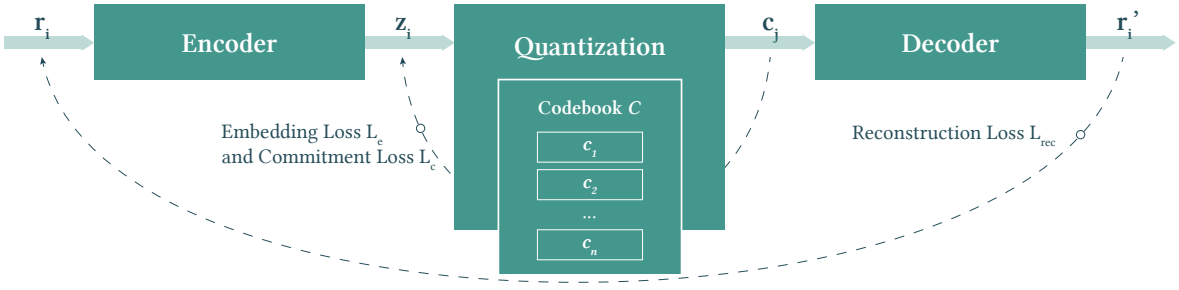


Fig. 2. We use a trained Encoder of a VQ-VAE as *Route Tokenizer* for *RouteLLM*. The VQ-VAE uses vector quantization to map inputs to discrete vector representations (codebook vectors). It is trained in a multi-objective optimization using a combination of multiple loss terms.

3.3 Route Tokenizer

Overview. Similar to unimodal LLMs, we train the tokenizer and the Transformer backbone separately. The tokenizer in our work is trained together with a corresponding decoder as part of a Vector Quantized-Variational Autoencoder (VQ-VAE) architecture. An illustration of this network is shown in Figure 2.

The VQ-VAE takes data like route information as input, compresses it into simpler, discrete vectors (codebook vectors), and then reconstructs the original data from those vectors. This allows us to map continuous route data to a manageable set of discrete tokens, which can be used to train an LLM to understand and reason about route context. The codebook vectors in the VQ-VAE can be considered semantic prototypes or primitives. Each vector in the codebook represents a distinct, meaningful aspect of the route context data, capturing core features such as road type, traffic level, or weather conditions.

The encoder E uses a route context vector r_i as input and encodes it to a representation $z_i \in \mathbb{R}^{d_e}$. This representation is then quantized by a vector-quantization layer Q to map z_i to a codebook vector c_j from a codebook C of size n with the lowest Euclidean distance to z_i. The decoder D then reconstructs the original input r_i from the quantized codebook vector c_j. The output reconstruction is denoted by r_i'.

Training. The VQ-VAE is trained by backpropagation of a multi-objective loss function that consists of three parts:

- Reconstruction Loss: This measures the difference between the input r_i and the reconstructed output r_i' and ensures that the decoder accurately reconstructs the original input from the quantized latent representation.
- Embedding Loss: This loss enforces proximity between the encoder's output and its nearest discrete codebook vector by minimizing the Euclidean distance between them. It ensures the latent representation is well-aligned with the pre-defined codebook, improving quantization.
- Commitment Loss: This regularizes the encoder by penalizing large deviations between the encoder's output and the selected codebook vector. It prevents the encoder from making arbitrary updates that could destabilize training by encouraging it to "commit" to the nearest vector in the codebook.

Since the quantization process is not differentiable, the gradients cannot be directly propagated from the decoder back to the encoder. To overcome this issue, we use straight-through estimators, i.e., we "copy" the gradient directly from the decoder to the encoder during the backward pass, thus bypassing the non-differentiable quantization step [8, 49]. As a result of the straight-through estimator, there is no gradient to optimize the codebook itself. Thus, following [49], we use an exponential moving average to update the codebook vectors instead of the embedding loss.

As opposed to previous applications of the VQ-VAE, the features of the input r_i in our work are heterogeneous, i.e., they comprise numerical as well as categorical features. For the numerical features, the reconstruction through the decoder represents a regression task, whereas for categorical features, it is either a binary or a multi-category classification task. To address this problem, we use a mixed reconstruction loss defined through

$$\mathcal{L}_r = \gamma_{cont}\mathcal{L}_{cont} + \gamma_{cat}\mathcal{L}_{cat} + \gamma_{bin}\mathcal{L}_{bin} \tag{1}$$

with weights γ_i and feature type-wise losses $\mathcal{L}_{cont}$ for continuous features, $\mathcal{L}_{cat}$ for multi-category features, and $\mathcal{L}_{bin}$ for binary features. During training, we select the correct loss function for each feature based on its type and then sum the losses for all feature types.

Due to the structure of the decoder network, the features encoded in the codebook vectors are jointly decoded into a reconstruction vector, which is used to calculate the loss. To facilitate the learning process and directly shape the embedding space with respect to each feature individually, we add feature-specific projection layers that predict the value of each feature from the output of the quantization layer. Due to the straight-through estimators used during the backpropagation of the loss function, these projection layers directly influence the output of the encoder and, therefore, regularize the embedding space. The loss function of the projection layers $\mathcal{L}_{proj}$ is a mixed reconstruction loss with the same loss weights as the decoder reconstruction loss. A coefficient γ_{proj} is used to control the influence of it.

Without appropriate regularization, the codebook vectors might not adequately represent realistic route context segments, leading to poor generalization and less useful route tokens for downstream tasks. Prior work on visual prototypes used r_1 and r_2 regularization to create meaningful and interpretable prototypes [31]. The r_1 loss leads to each codebook vector being as close as possible to the input vectors, making them interpretable since the loss anchors the codebook vectors in the training dataset. The r_2 loss is equal to the embedding and commitment loss calculated over a whole batch. We include the r_1 loss to increase the interpretability of our codebook vectors since we want each of the resulting route tokens to be close to realistic route segments. It is defined as

$$\mathcal{L}_{r_1} = \frac{1}{n}\sum_{j=1}^{n}\min_i \|\mathbf{z}_i - \mathbf{c}_j\|_2^2. \tag{2}$$

Finally, we add L_1 regularization on the codebook for sparsity, i.e., to ensure that only a small subset of the codebook vectors' entries are activated during encoding. By encouraging sparsity, we improve the model's efficiency and enhance the interpretability of the route tokens, as each activated codebook vector will correspond to more distinct and meaningful route context features. Thus, the full loss function for training the route tokenizer is given through

$$\mathcal{L} = \underbrace{\gamma_{cont}\mathcal{L}_{cont} + \gamma_{cat}\mathcal{L}_{cat} + \gamma_{bin}\mathcal{L}_{bin}}_{Reconstruction} + \underbrace{\lambda_{L_1}\|C\|_1 + \lambda_{r_1}\mathcal{L}_{r_1}}_{\text{Regularization}} + \beta\mathcal{L}_c + \gamma_{proj}\mathcal{L}_{proj} \tag{3}$$

3.4 Transformer Backbone

Overview. We use the trained *Route Tokenizer* and add it as an additional tokenizer to the Transformer backbone of an LLM, pretrained on natural language. Thus, we add the route tokens and three new special tokens to the vocabulary of the Transformer backbone. We use the codebook vectors' indices as route tokens, e.g., `[route:0000]`, and add them to the vocabulary. Additionally, we add a padding token, a beginning of route token `[ROUTE]`, and an end of route token `[/ROUTE]`. We extend the embedding table and the weight matrix of the output linear layer of the model to match our new vocabulary size. To initialize the embedding table for the newly added route tokens, we scale the corresponding codebook vectors to the order and magnitude of the text embeddings in the embedding table and initialize the route context embeddings with it. For initializing the entries of the output

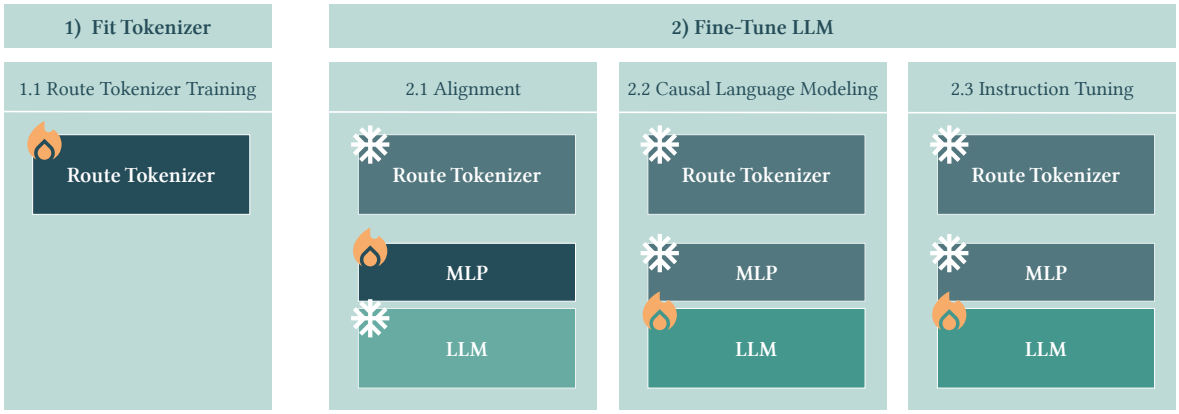


Fig. 3. Our whole Training Pipeline consists of four steps. First, we train the *Route Tokenizer* as the encoder of a VQ-VAE (see figure 2), and attach it to a pretrained LLM. Then we train an MLP to project the route context embeddings into the same subspace as the language embeddings, followed by fine-tuning the LLM for cross-modal understanding. Finally, we use an instruction tuning step to improve route-context understanding and reasoning.

linear layer, we follow [24] and sample the new weights from a normal distribution with the mean and covariance matrix of the original weights.

Due to the architecture of the VQ-VAE, embeddings of similar route tokens have a low Euclidean distance. On the contrary, embeddings of semantically similar text tokens tend to have a high cosine similarity. To account for these different subspaces, we add an MLP projection network after the route embedding table to align the embeddings of the text tokens with the embeddings of the route tokens. The MLPs architecture follows [33].

Training. Building upon [28], we train the LLM for cross-modal understanding between routes and natural language in three steps (see Figure 3) using multimodal datasets:

(1) **Alignment of Route and Text Tokens:** First, we train the embedding projection layer to learn a projection from the modality embedding space, in this case, the codebook embedding vectors, into the LLM embedding space [15, 34]. Thus, we freeze all model parameters except those in the embedding projection layer and train the model using causal language modeling to "translate" route tokens to natural language.
(2) **Multimodal Causal Language Modeling:** Next, we freeze the projection layer's parameters and train the LLM's parameters using causal language modeling (CLM), also known as next token prediction, on a dataset of routes paired with textual descriptions. These natural language route descriptions are often sparse, i.e., they refer only to a small subset of features for a small portion of the tokens per route. This makes it challenging for the LLM to accurately attribute specific features to specific tokens within the route. To simplify this attribution, we train the model to generate a feature description based on a single route token at a time. This approach resembles a regression or classification task, depending on the feature value the model is predicting.
(3) **Multimodal Instruction Tuning:** Finally, we fine-tune the LLM's parameters on several route-related tasks to improve its route-context understanding and cross-modal understanding. These tasks include the generation of route descriptions or traffic reports, the sorting of route segments into chronological order, and the interpolation of routes. Following [51], we add Chain-of-Thought (COT) examples to increase the generalization capabilities, improve robustness, and improve the zero-shot reasoning capabilities of the

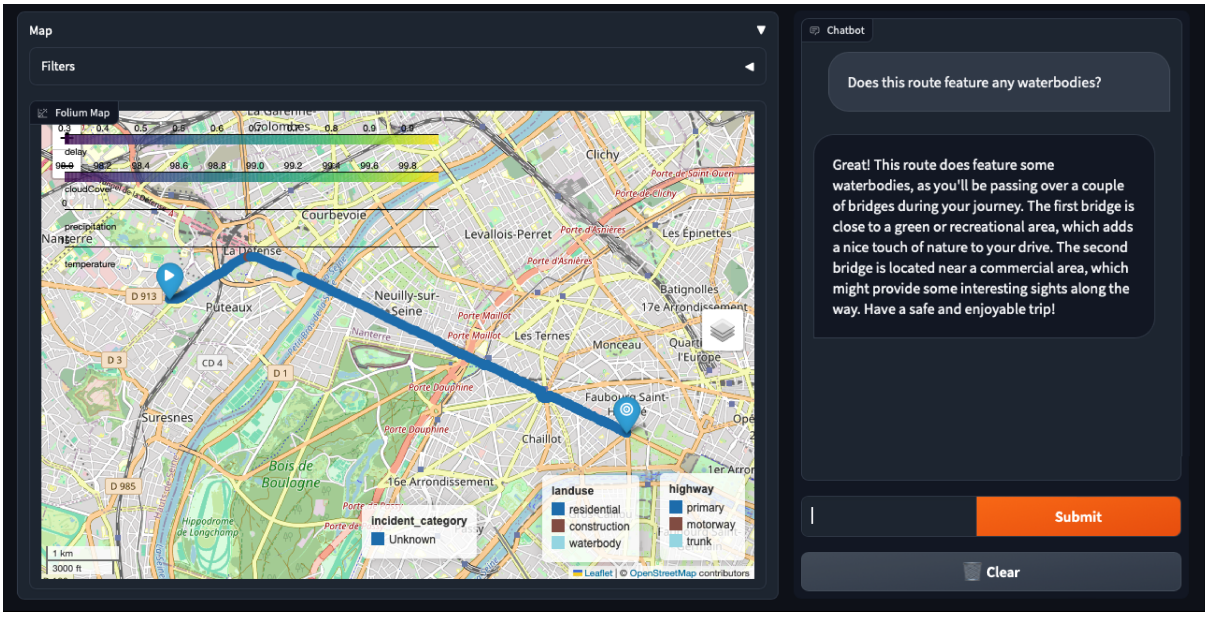


Fig. 4. We build a Web Interface for *RouteLLM*, to enable easy and accessible interaction with the system. In the interface, the user can select a route, and then interact with it via a chat interface.

model. We add COT triggers, such as *Let's think step by step*, into the tasks, that force the model to split them into smaller subtasks following the logical reasoning steps.

4 IMPLEMENTATION

4.1 Route Tokenizer

Following [28], we use residual networks as the Encoder and the Decoder networks of the VQ-VAE. The encoder consists of a series of $\lfloor \log(\frac{m}{2}) \rfloor$ encoder blocks, each cutting the dimension of their input in half. The encoder blocks consist of a 1D convolution and three residual blocks. The residual blocks consist of two 1D convolutions, a batch normalization (BN) layer, and a ReLU activation function. All convolutional layers, except the last one, output a feature map of d_h channels. The last convolutional layer of the encoder outputs d_e channels. The decoder consists of a series of $\lfloor \log(\frac{m}{2}) \rfloor$ decoder blocks, similar to the encoder blocks but in reverse. They consist of 3 residual blocks with dilation rates 9, 3, and 1, an upsampling layer that scales the input dimension by 2 using nearest neighbor interpolation, and a 1D convolution. The last convolutional layer of the decoder outputs 1 channel. We use a smooth L1 loss as $\mathcal{L}_{cont}$, a categorical cross entropy loss as $\mathcal{L}_{cat}$, and a binary cross entropy loss as $\mathcal{L}_{bin}$. To address imbalances in the categorical features, we use class weights and label smoothing [46] in the categorical and binary cross entropy loss. Finally, we use codebook resets for improved codebook utilization [40], i.e. we periodically reinitialize a subset of the codebook vectors based on the distribution of the currently observed input vectors.

4.2 Transformer Backbone

Base Model. We use a decoder-only architecture as the backbone of *RouteLLM* due to the recent popularity gain of decoder-only LLMs and their widespread usage as chatbots, allowing us to build *RouteLLM* based on a wide

range of pre-trained models. Specifically, we chose the instruction-tuned version of Mistral 7B [27]. The model shows promising performance on LLM benchmarks, such as MMLU [22], HellaSwag [55] and WinoGrande [43], and its comparably smaller size allows for local inference on consumer devices. We additionally reduce the number of parameters and their memory consumption using Quantized Low Rank Adaptation (QLoRA) [19, 26], with the rank set to $r = 16$ and a scaling factor of $\alpha = 16$. After attaching the *Route Tokenizer* and extending the model's vocabulary, it comprises 33 027 tokens, 32 000 of which are text tokens, 1 024 are route tokens, and 3 are special tokens (one route padding token, one begin of route token, and one end of route token).

Training. We train the projection layer for aligning the route token embeddings with the text embeddings for 10 epochs using a batch size of 8 and a gradient accumulation of 16 iterations. We use a cosine learning rate schedule with an initial learning rate of $1e-3$ and a warm-up period of 50 iterations. Next, we train the model for 1 epoch using QLoRA in the Multimodal Causal Language Modeling step. We use a batch size of 8, a gradient accumulation of 16 iterations, and a learning rate of $2.5e-5$ with a cosine schedule considering the first 3 % of the iterations as warm-up period. Finally, we instruction-tune the model for 15 epochs using a learning rate of $2.5e-5$ with a cosine schedule and a warm-up period of 3 % of the iterations. We set the batch size to 4 and accumulated the gradients over 8 iterations. Again, we only train the QLoRA adapters in the LLM.

4.3 Training Details

We implement the *Route Tokenizer* and *RouteLLM* using PyTorch [4] and train it with Hugging Face Transformers [52] on an Amazon Web Service's `g5.12xlarge` instance, equipped with four Nvidia A10G GPUs with 24GB VRAM each. The alignment step took on average 34 hours, the Multimodal Causal Language Modeling step took on average 45 hours, and the Instruction Tuning took on average 25 hours. We use QLoRA and FlashAttention2 [17] in all training stages except the alignment stage. We select the best model based on the validation score for each training stage. Finally, the model is unquantized, by fusing the QLoRA adapters storing the parameters in half-precision.

4.4 Interface for Inference from User Input

To use *RouteLLM* with user input, we create a web-based user interface using Gradio [1] and serve the trained model using FastChat [57]. Figure 4 shows a screenshot of the interface. First, users enter a custom start and endpoint for a route. The system will then automatically calculate the shortest route between these two points as a sequence of edges in a street graph. When using *RouteLLM* in an in-the-wild application, this data would be typically obtained from real-time GPS sensors and a navigation system. The edges are then annotated with the context features necessary for *RouteLLM* by retrieving the data from web APIs. In a real-world application without internet access, the data could also be obtained through locally hosted APIs that store a contextualized road graph, or inferred from camera images. Still, it has to be assured that the utilized feature values, such as the weather categories (e.g., *cloudy* or *sunny*), align with the ones that were used for training *RouteLLM*. Additionally, missing feature values might need to be filled, e.g., through forward-filling. Finally, the complete sequence of route context is tokenized through our *Route Tokenizer*. Users can then start to ask questions on the route and leverage *RouteLLM*'s route context reasoning capabilities.

5 DATASETS

We propose two datasets for training the *Route Tokenizer* and *RouteLLM*: *ContextualRoutes* and *LabeledRoutes*. In the following, we present the creation strategy and their details.

5.1 Context Feature Selection

For creating the datasets, we need to describe route context through a set of feature values. These features could include data from various sources describing diverse aspects of the scenes, e.g., road states, weather information, or the landuse around the location. Further, these features can either vary over time (dynamic context) or be constant (static context). In order to prevent a costly, laborious, and irreproducible data collection process, we opt to include only features that can be retrieved through online APIs or derived from these. Such features include elevation, the incline of a road segment, the traffic situation, or the expected travel along a road segment. Further, we refrain from using images, e.g. retrieved from StreetView databases, for three reasons. First, using image data limits the generalization ability of a model trained on its basis, as large areas on Earth are not covered with this data, and some covered areas might be incompatible as the cameras used to capture the images differ too much from the ones used to capture the images in the training dataset. Second, it comes with significant privacy issues as images may capture identifiable individuals, vehicles, or private properties, raising ethical and legal concerns regarding data usage. And third, processing and storing large-scale image data is computationally expensive, requiring substantial resources for both storage and model training. Thus, we decide to include numerical, categorical, and boolean features from four clusters: temporal features, weather features, road features, and traffic features.

5.2 *ContextualRoutes*

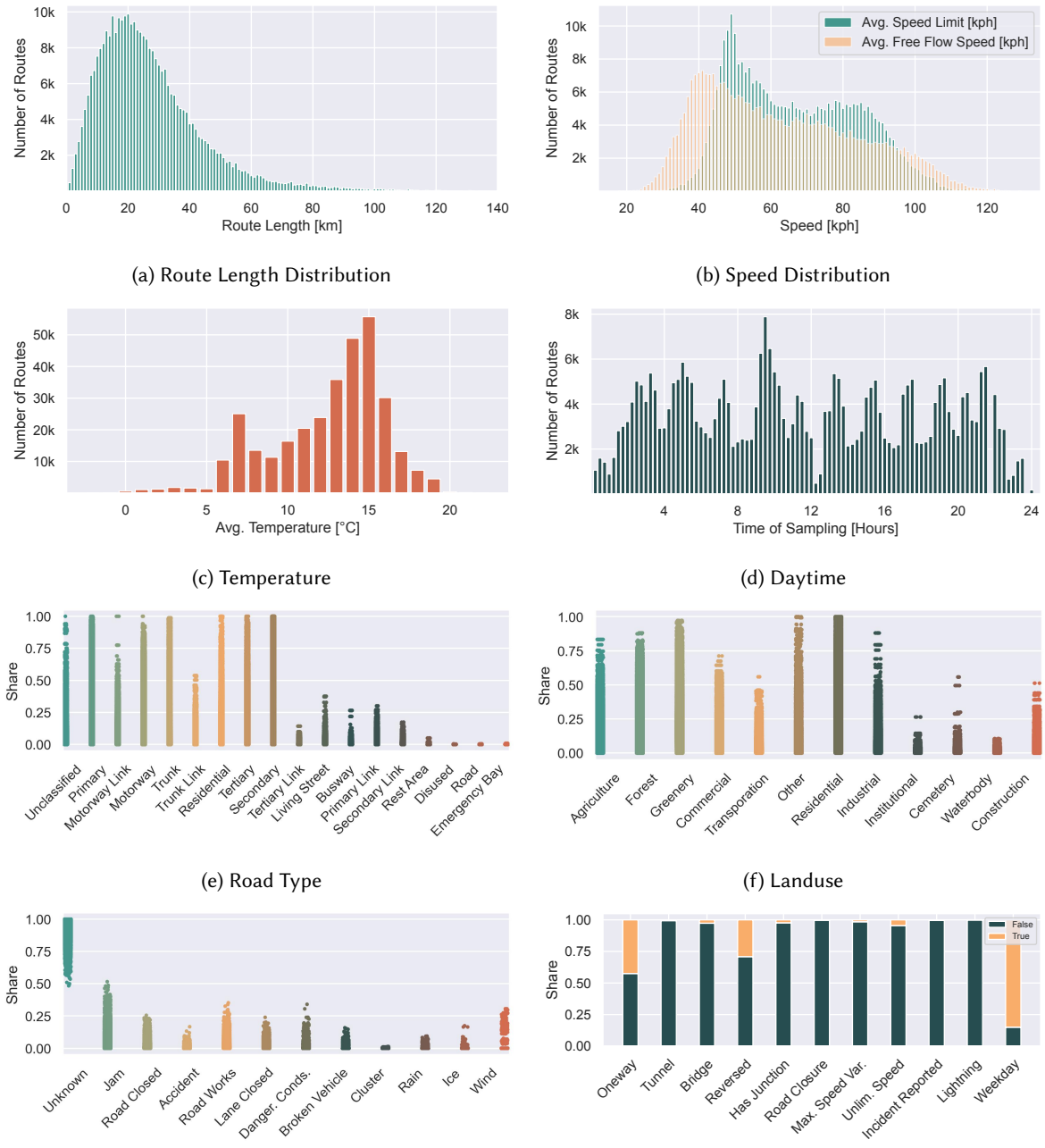
For training the *Route Tokenizer*, we require a dataset that comprises a variety of route context features and is sufficiently large to train a VQ-VAE. To these means, we first sample a fixed set of routes, annotate it with static context features, and then enrich them with dynamic features several times. This allows us to capture the same route with potentially different dynamic feature values at different times.

We define 18 bounding boxes from which we sample the routes (see Figure 11 for an overview and Table 2 for the precise positions). The bounding boxes include the ten biggest cities in Germany, two rural regions in Germany, and four European cities outside Germany. Their areas range from approximately 750 km^2 to approximately 17800 km^2. For each bounding box, we retrieve a road graph from OSMnx [13], add elevation data from the Japan Aerospace Exploration Agencies (JAXA) AW3D30 global digital surface model[4], and calculate the Menger curvature and heading for each edge. Additionally, we sample a map annotated with usage areas (residential areas, farmland, or construction sites) for the same bounding box using the OSM landuse key, and assign each edge in the road graph the landuse of the area it is closest to.

From this contextualized street graph of each location, we randomly sample a set of nodes, divide them into start and destination nodes, and compute the shortest route between each start and destination node. To address the potential issue of resulting routes being too similar, we filter out redundant routes using the Hausdorff distance [12, 42]. The Hausdorff distance between two routes is defined as the maximum Euclidean distance from any point on one route to the closest point on the other. After this filtering process, for Stuttgart, only 0.02% of the Hausdorff distances between two routes are less than 1km, and on average, there is only one other route with a Hausdorff distance lower than 1km per route. Additionally, we discard all routes with a total length of less than 1 kilometer.

We randomly select these contextualized routes at arbitrary times across the day and add dynamic features from Azure Maps through map tiles[5]. These dynamic features include traffic flow, road incidents, and weather information. Since we use a different underlying map provider than Azure Maps, we match our route segments with the route segments in the tile. We use a buffer of 5 meters around each route segment for the matching process. The traffic flow tiles include the feature `delay`, which indicates the fraction of the free flow speed at

[4] https://www.eorc.jaxa.jp/ALOS/en/dataset/aw3d30/aw3d30_e.htm, last accessed July 16, 2025
[5] https://learn.microsoft.com/en-us/rest/api/maps/render/get-map-tile, last accessed July 16, 2025



(a) Route Length Distribution

(b) Speed Distribution

(c) Temperature

(d) Daytime

(e) Road Type

(f) Landuse

(g) Traffic Incidents

(h) Various Binary Features

Fig. 5. We visualize the distribution of selected features from the *ContextualRoutes* dataset. The plots demonstrate the variety of route context present in the dataset.

which the traffic is currently moving. We use this value to calculate the current traffic speed and the current travel time along each route segment. Based on these travel times, we then retrieve the weather at each route segment for the estimated time of arrival. We sample the weather every 250 meters to reduce the number of requests, and save the current server time and date as the route timestamp. Finally, we encode the local time as sine and cosine components in order to account for the periodicity of the signal [30].

The final *ContextualRoutes* dataset consists of 324 802 routes with a total length of 8 915 625.3 km. Each route segment is annotated with 39 context features, 14 of which describing the road state, 3 describing temporal aspects, 7 describing the weather, and 15 describing the traffic. We provide an overview of all features with a description in Table 3 and visualize the distribution of selected features in Figure 5.

5.3 *LabeledRoutes*

While *ContextualRoutes* comprises routes annotated through context features, we additionally require a multimodal dataset containing both routes and natural language to train *RouteLLM* for cross-modal understanding. For this, we build upon the *ContextualRoutes* dataset and annotate the routes with labels using two different strategies. First, we use a template-based automated annotation for simple tasks; second, we use a teacher model to generate labels that require reasoning based on the route context.

Templates are particularly effective for straightforward questions about route characteristics, like the length of a route. For example, to generate a response for the question *How many bridges are along this route?*, we evaluate the correct label value based on the underlying route and insert it into a label template, e.g., *There are [*n*] bridges along this route*. To increase the diversity of samples, we create multiple templates per question and answer. The features the questions refer to are randomly selected, except for categorical features with a strong class imbalance. For example, when dealing with highways, the selection process adjusts for feature variability, as categories like 'highway' could otherwise oversample the answer 'no' for rare classes.

Tasks that involve reasoning, diverse language, or feature abstraction cannot be annotated using simple templates. Therefore, as a second strategy, we use a teacher model to annotate the routes with diverse natural language labels for such tasks, e.g., evaluating route beauty or summarizing weather conditions. The teacher model is prompted with style guidelines for the response phrasing and a brief description of the route features. By varying the style instructions, we increase the diversity of the generated labels. We query the teacher model with a randomly selected question and a route description generated through a template that describes the features of the route segment by segment. As we found the teacher model tending to mirror the style of the route descriptions (probably due to their length), we limited its access to detailed information to generate more natural responses. For instance, instead of providing exact numerical values for precipitation, we categorize the precipitation levels into broader natural language terms.

We form three suitable datasets, one for each fine-tuning step of *RouteLLM*, based on *ContextualRoutes* and the annotation strategies introduced above.

(1) **CausalSingleToken** This dataset uses template-annotated labels and is used for the first training stage, i.e., the alignment of the route and text tokens. For each token in the route vocabulary, we first randomly sample 500 route segments from the *ContextualRoutes* dataset that are represented by it and then randomly select a feature. Then, we annotate the segment with a template-based label utilizing the selected feature's value, e.g. *"The travel time is 0.7 s"*. In total, this dataset contains 512,000 samples, out of which 407,212 are used as training samples, and 104,788 are used as validation samples.
(2) **RouteInstruct**: We create a multimodal, multi-turn (i.e. multiple consecutive utterances per conversation), multi-task dataset for instruction tuning using a mixture of template and teacher annotated labels. An overview of the corresponding tasks is shown in Table 4. The final dataset comprises 11 936 conversations

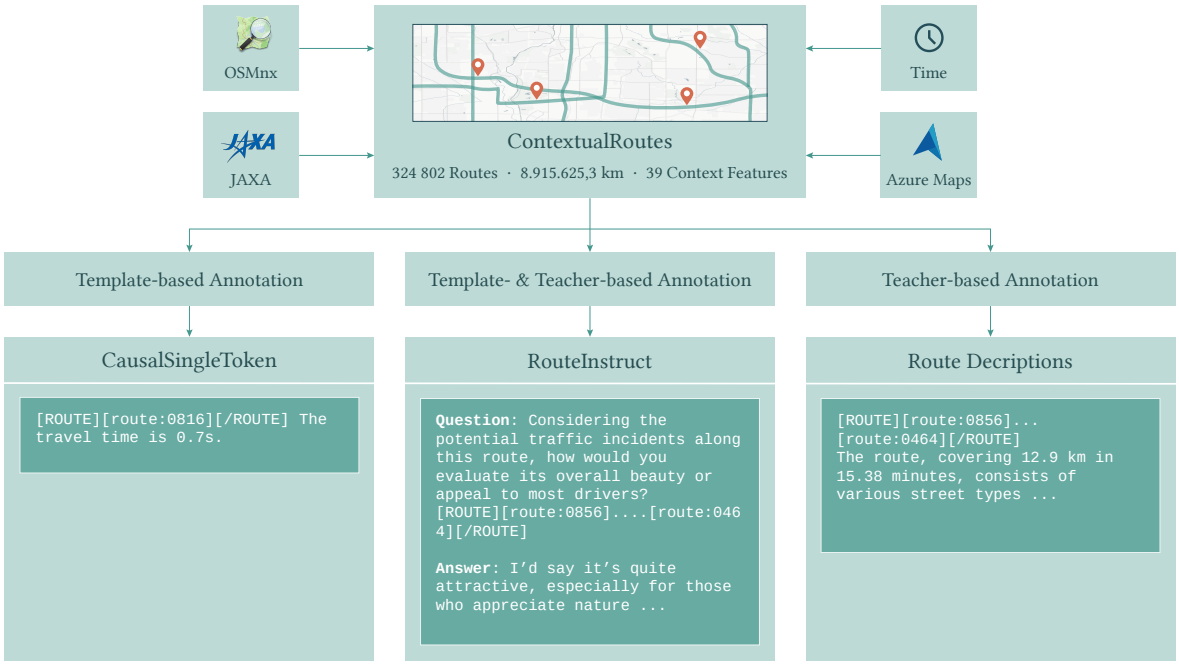


Fig. 6. We annotate the routes of the *Contextual Routes* dataset with multiple labels to create the *LabeledRoutes* dataset. The associated tasks require three different sets of skills: feature extraction, route reasoning, and route understanding.

consisting of 16 052 turns, of which 5 470 require feature extraction, 1 748 require route understanding, and 8 834 require route reasoning.

(3) **Route Descriptions**: Finally, we create a route question-answering dataset using the textual descriptions from RouteInstruct. All of the samples in the dataset are annotated using a teacher model. The dataset differs from the CausalSingleToken dataset since it requires the model to predict the feature values of an entire route, thereby requiring the model to condense the description into a few sentences, and thus forcing the model to extract essential route characteristics since it cannot describe every feature value of each token. It contains 6,561 samples, of which 4,262 are training samples, 875 are validation samples, and 1,412 are test samples.

5.4 Generalization of the Dataset to Other Regions on Earth

Since *RouteLLM* relies on route context tokens learned during training the *Route Tokenizer*, its ability to generalize to unseen locations depends on the diversity of feature distributions in our dataset. Thus, to create the datasets, we sampled routes from 18 different bounding boxes across Germany, France, Great Britain, and Iceland. These bounding boxes comprise a wide range of landscapes, from urban city centers to rural roads and mountainous regions. Additionally, this accounts for varying traffic patterns, including roundabouts, large intersections in high-density metropolitan areas, and windy rural routes in sparsely populated areas to capture diverse contextual features. To assess if the routes cover the necessary feature variance, we visualize the feature values and their combinations in Figure 5. Our analysis reveals long-tail distributions, with feature values spanning nearly the entire range of possible values defined by OpenStreetMap. While this indicates a broad coverage of contextual features within the sampled regions, we acknowledge that the dataset lacks representation of certain global areas

such as Africa or Southeast Asia, which may show different feature distributions, e.g., for the temperature feature. To facilitate future extension of the dataset, we released the source code used for sampling the routes[6]. Within the current geographic scope, however, the diversity of route features is sufficient for the *Route Tokenizer* to learn tokens for a wide range of value combinations, potentially enabling generalization to unseen but structurally similar regions beyond the original bounding boxes. To validate this in the following model evaluation, we used routes sampled from different bounding boxes for testing than those used during training.

6 EVALUATION

We evaluate to what extent *RouteLLM* learns a cross-modal understanding following our design and training strategy. We first evaluate the *Route Tokenizer* regarding its ability to encode route segments into semantically meaningful embeddings and thus provide rich input to *RouteLLM*. Then, we evaluate *RouteLLM* as an end-to-end system.

6.1 *Route Tokenizer*

Route Tokenizer's capability to learn semantically meaningful and useful codebook vectors is determined through the following two criteria:

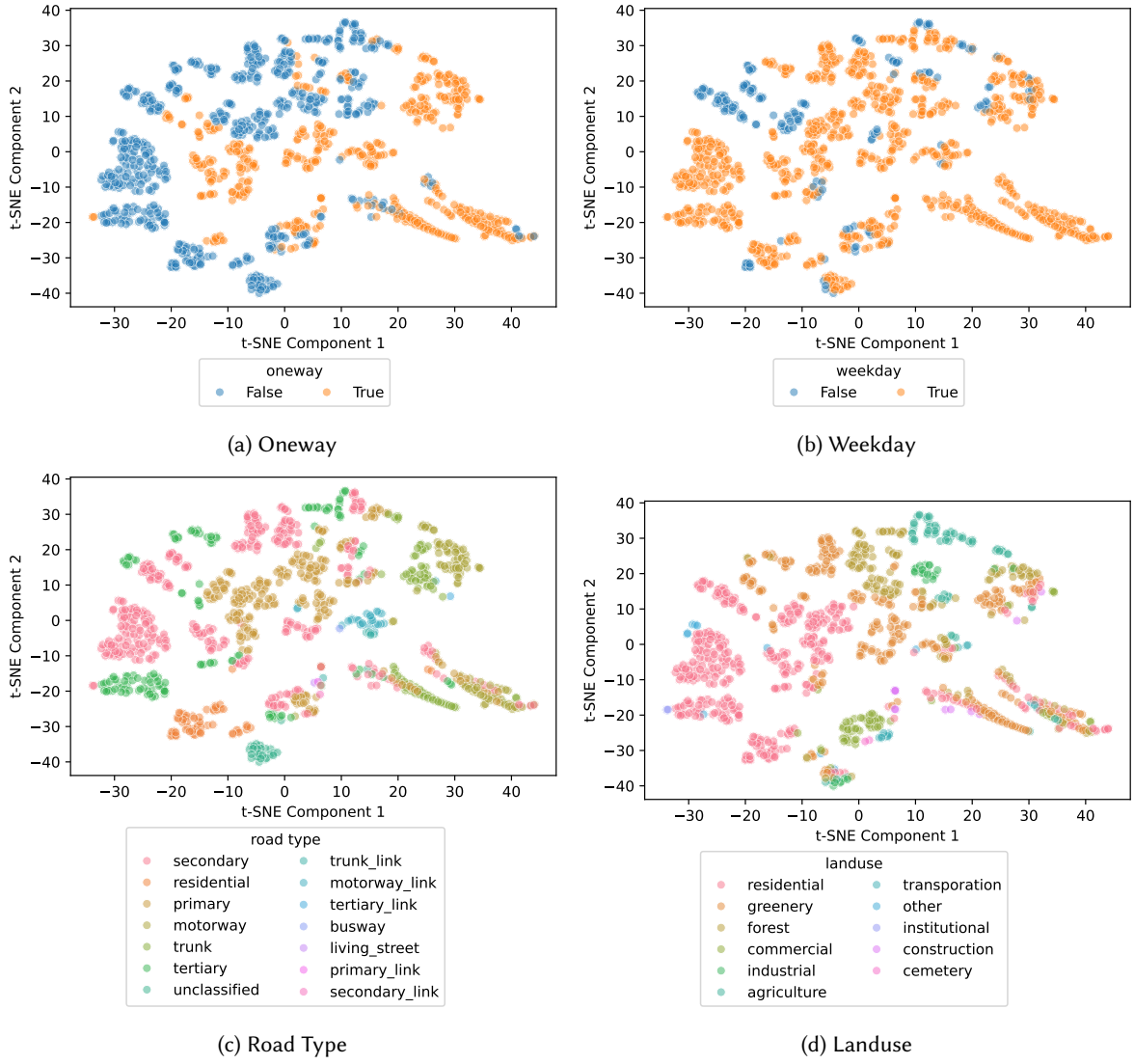
- C1 **Feature Clustering:** Segments with similar features should be close to each other in the codebook space
- C2 **High Feature Expressiveness:** The codebook should have a high expressiveness regarding route context features to cover the entire feature space, e.g, there should be at least one codebook vector per category for categorical features.

We start by visually inspecting the codebook vectors learned by the *Route Tokenizer* and then quantitatively analyze the encoding and reconstruction process.

6.1.1 Qualitative Analysis. We reduce the number of dimensions from the codebook vectors to 50 using a principal component analysis (PCA) and then reduce the remaining dimensions to 2 using a t-distributed stochastic neighbor embedding (t-SNE) [50]. Figure 7 visualizes the t-SNE embedded codebook vectors colored by different features. We observe that the values of the categorical feature `road_type` distinctly separate the clusters in the codebook space. The same applies to the `landuse` feature, where only one type of landuse is assigned to most clusters, leading us to conclude that they are the main distinguishing features for the model. We attribute this cluster structure to the high weight of the categorical features in the loss function. Binary features, such as the `oneway` and `weekday` features, seem clustered within these bigger clusters. Interestingly, all traffic incidents are located in one cluster, with road types in this cluster mostly being roads outside of cities, such as motorways or trunk roads. Additionally, the `delay` feature value of most codebook vectors in that cluster shows an apparent reduction of the traffic speed compared to the free flow speed. The codebook vectors in that cluster indicate a high chance of incidents and a high incident magnitude. Thus, *Route Tokenizer* can learn the correlation between such traffic features. On the other hand, we find some categories missing. For example, none of the codebook vectors represents bridges, tunnels, thunderstorms, or variable maximum speed, leading to a loss of expressiveness. Similar problems can be observed for numerical features, such as the temperature or the precipitation. The range of the recorded weather phenomena is limited in the dataset because the recording only spans over one week in comparatively few regions, leading to a biased dataset. As a result, the `temperature` feature encoded through the codebook vectors ranges between approximately 3 and 17 degrees Celsius. The times range from 9 a.m. to 6 p.m. on a weekday, resulting in the model being unable to express a traffic incident on the weekend or at night.

6.1.2 Quantitative Evaluation. Next, we evaluate the VQ-VAE's performance by its ability to reconstruct the original feature values of the route segments from the testset of *ContextualRoutes*. We report the mean average

[6] https://github.com/verena-hallitschke/routellm



(a) Oneway (b) Weekday

(c) Road Type (d) Landuse

Fig. 7. Visualization of the Codebook Vectors through a t-SNE Embedding colored by different features.

error (MAE) for numerical features in Table 1 and show confusion matrices for binary and categorical features in Figure 12. Overall, the error tends to be moderate or high for most features, while weather and traffic incident features show a lower error value. Further, we find that the model cannot precisely reconstruct the daytime but can estimate a range (morning, noon, evening, and night). We also find the model to be incapable of reconstructing features that describe the segment's geometrical structure, such as the grade or heading.

We find this due to the quantization significantly reducing the number of possible values that can be encoded through the codebook vectors for each feature. For correlated features, the codebook vectors only need to store a small subset of all possible value combinations without losing expressiveness. Thus, the higher the correlation of

Table 1. We report the VQ-VAE reconstruction for numerical features as Mean Average Error (MAE) for the three feature categories time and weather, traffic and road properties.

Time and Weather		**Traffic**		**Road Properties**	
Feature	MAE ↓	Feature	MAE ↓	Feature	MAE ↓
cos_time	0.4	current_speed	19.9 kph	curvature	0.073 m^{-1}
sin_time	0.31	current_travel_time	1.5 s	grade	0.043
time	2.65 h	delay	0.056	heading	90°
cloudCover	15.4 %	free_flow_speed	12.9 kph	landuse_distance	5.8 m
precipitation	7.5 dBZ	incident_certainty	0.0083	lanes	0.45
temperature	2.1° C	incident_delay	2.5 s	length	20.27 m
windGust_speed	6.3 kph	incident_distance	0.006 m		
wind_direction	39°	incident_magnitude	0.004		
wind_speed	3.4 kph	speed_kph	12.4 kph		
		travel_time	1.2 s		

two features, the more distinct values can be represented in the codebook space with the same number of codebook vectors. On the other, the codebook vectors should encode all possible feature combinations for uncorrelated features such as `length` and the `precipitation`. We argue that due to the high correlation of the features in the weather and traffic incident categories, their effective feature resolution is significantly higher than those in other categories. Accordingly, we assume that the model's performance on binary and categorical features is superior to that on numerical features due to their already limited value space. Supporting this hypothesis, we find the reconstruction quality to depend on the number of feature categories in the training dataset. For example, the prediction accuracy is high for features with an approximately balanced number of samples per category, e.g. `oneway`, with true-positive rates of consistently over 80 %. On the other, features with strong class imbalances, e.g. the `bridge` feature, suffer from worse reconstruction.

6.1.3 Ablations. Next, we present ablation studies of the most important hyperparameters of the *Route Tokenizer*.

Codebook Dimension. Increasing the codebook dimension impacts the amount of information the codebook vectors can encode and, therefore, the amount of information it can pass to the decoder. Using smaller codebook vectors leads to a smaller model size and more efficient processing. However, it can also lead to a loss of information and, therefore, worse decoding performance. We compare the validation reconstruction loss between models using a codebook dimension of 64, 512, and 4 096. We find that increasing the codebook dimension from 64 to 512 leads to a slight performance gain, while the increase from 512 to 4096 leads to a slight performance decrease. Still, we decide to use codebook vectors of size 4096, due to the base model used for *RouteLLM* (Mistral) using embeddings of this size. Matching the codebook dimension of the *Route Tokenizer* with the embedding dimension of *RouteLLM*, allows us to use the codebook vectors directly as token embeddings.

Codebook Size. The influence of the quantization heavily depends on the number of codebook vectors. An increase in codebook vectors increases the values the codebook space can represent, thus potentially adding representations of undersampled classes that could not have been expressed before. In turn, the increase might also lead to more representations of oversampled classes, thus resulting in a codebook space very dense in areas with high dataset frequency and sparse in regions with low dataset frequency. We compare the validation performance of the model using 512, 1 024, and 2 048 codebook vectors. We find that the model collapses when using 512 codebook vectors. Further, we find that increasing the number of embedding vectors slightly improves

the model's predictive performance. In our experiment, the model introduces codebook vectors for the `bridge` feature when increasing the number of codebook vectors to 2 048, indicating that increasing the number of codebook vectors could improve the prediction performance on undersampled categories. On the other, an increase in codebook vectors implies an increase in route tokens, making subsequent training of an LLM with these tokens more complex. We decide to use a codebook with $1,024$ codebook vectors, trading off predictive performance with embedding space redundancy and LLM training complexity.

Hidden Dimension. A higher hidden dimension results in a wider VQ-VAE network, thus increasing the risk of overfitting, while a lower hidden dimension increases the risk of underfitting. We compare the model's validation performance with hidden dimensions of 16, 32, 64, and 128. The hidden dimension increases the validation performance on binary and categorical features, with the binary feature performance showing the biggest performance gain. Notably, with a hidden dimension of 128, the model learns embeddings of features it could not learn with the lower dimensions, e.g., embeddings for the `bride` and `tunnel` features. There is no significant change in the prediction performance for numerical features.

Auxillary Losses. We use an L_1 regularization loss on the codebook to encourage sparsity, as we expect this to lead to more specialized and diverse codebook vectors. However, this increased specialization comes at the risk of reduced codebook utilization. Thus, we compare the effect of three different regularization techniques:

A: No regularization, i.e. neither L_1 regularization nor r_1 and r_2 losses
B: regularization through r_1 and r_2 losses
C: L_1 regularization and r_1 and r_2 losses

We find the precision and recall of the binary features to decrease in both scenarios with reduced regularization (A and B). On binary features, the recall decreases from 0.68 (C) to 0.58 (B) and 0.60 (A) respectively, and the precision decreases from 0.60 (C) to 0.54 (B) and 0.56 (A). We do not observe an effect of regularization on the numerical features.

6.1.4 Conclusion. In summary, we find the codebook vectors to be meaningful for most features, while their expressiveness is limited due to missing feature values and missing value combinations for some features. Specifically, we find the route tokens do not encode information about bridges, tunnels, or whether the maximum speed is variable. Further, they encode only coarse time, traffic flow, and road geometry information. On the other, the route tokens encode fine-grained information regarding weather, traffic incidents, road types, and the direction of traffic. Thus, based on the prediction results on the test set and the visualization of the codebook space, we conclude that even though not optimal, the learned codebook vectors are of sufficient quality for our proof of concept to train the first route context-aware LLM.

6.2 *RouteLLM*

We evaluate *RouteLLM's* generations both manually and automatically. For the manual evaluation, we randomly select routes from the training and validation set and experiment with diverse tasks from the RouteInstruct set as well as new and unseen tasks. Since manual evaluation is slow, we additionally use gpt-4o[7] to automatically evaluate the model's generations on RouteInstruct's validation dataset.

6.2.1 Manual Evaluation. When interacting with *RouteLLM*, we find that the model can understand most questions and generate fluent answers. We present exemplary conversations with *RouteLLM* in Figure 9. In the process of learning cross-modal understanding between text and routes, it seemed to keep its world knowledge since it is still able to answer general questions such as *What is the Pythagorean theorem?* or understand foreign

[7] https://openai.com/index/hello-gpt-4o/, last Accessed July 16, 2025

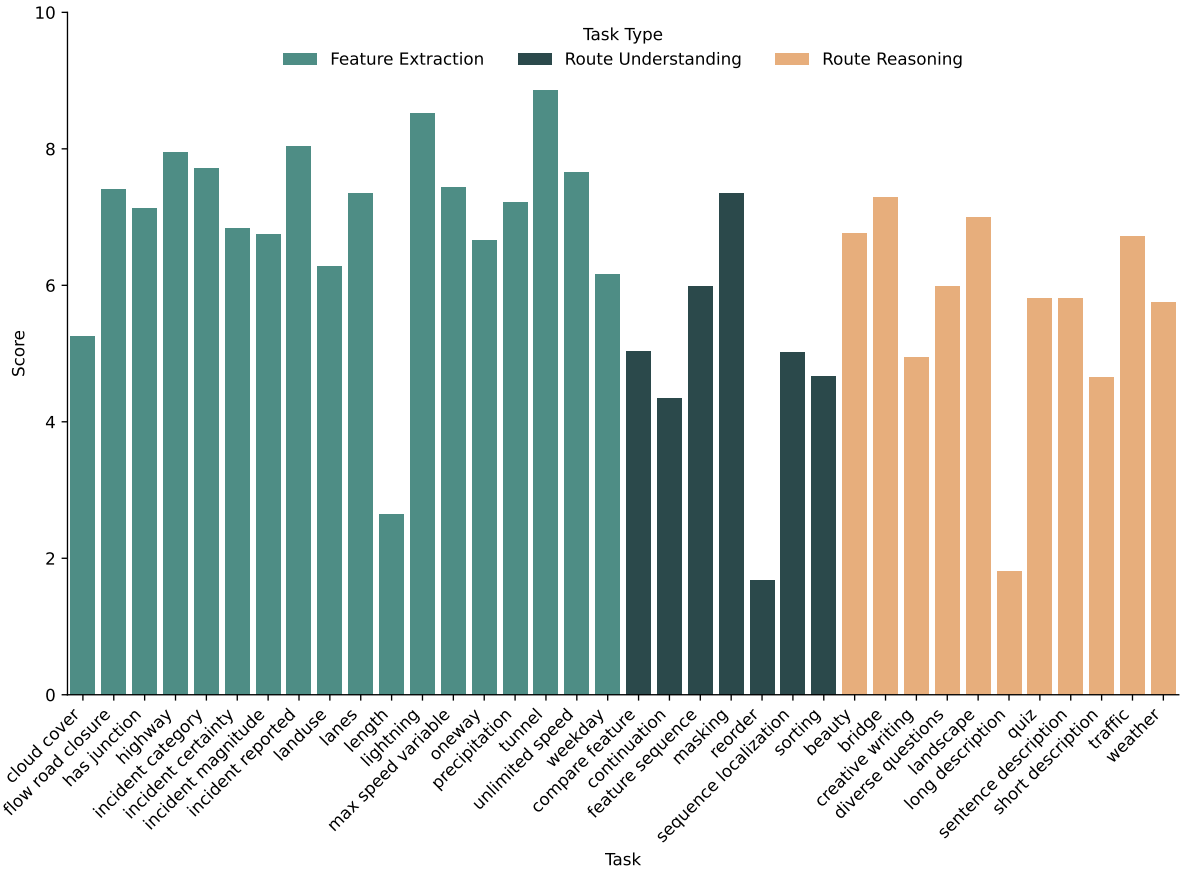


Fig. 8. Evaluation for the different Tasks in *LabeledRoutes*

languages, even though *LabeledRoutes* only contains English samples. Moreover, *RouteLLM* demonstrates zero-shot reasoning capabilities on route context data, for example, by reasoning whether a route is suitable for noise-averse passengers, a question not represented in the training dataset.

6.2.2 LLM-based Evaluation. Next, we evaluate the generations of *RouteLLM* using gpt-4o. Since *RouteLLM* is instruction-tuned with a maximum route length of 256 tokens, we only include samples from the validation dataset within this limit. Following [57], we instruct gpt-4o to score each generation on a scale of 0 to 10. We provide the overall scores and an overview of the mean scores per feature in Figure 8. The results of the automatic judgment generally align well with our findings when interacting with the system. The mean score across all task categories is 6.23, with the mean score for the feature extraction tasks slightly above the total average at 7.02. The scores of the Route Reasoning and Route Understanding tasks are slightly lower at 5.97 and 4.90, respectively. When comparing the gpt-4o scores, the tasks with the highest scores are the `tunnel-task`, `lightning-task`, and `incident-reported-task`. We assume that the class imbalance of those features leads the model to predict a constant value for those features while still receiving a high score. On the other, the `length task`, `long description task`, and the `reorder task` rank the lowest. Solving the `reorder task` requires strong route

Human: In one sentence. Is this route beautiful?
RouteLLM: sure, it's mostly flat, has some green spaces and a few residential areas along the way, so it could be quite scenic. Just remember to drive carefully and enjoy the ride!

Human: Is it cloudy?
RouteLLM: Great question! It seems like the weather is going to be quite pleasant for your journey. There's no rain expected, and the sky will be mostly cloudy with a temperature of around 17.4 degrees Celsius. The wind will be blowing at around 8-9 kph, with gusts reaching 13 kph. So, you should have a comfortable trip without any rain to worry about.

Human: What is the vibe of the route?
RouteLLM: Great question! The vibe of the route you've described seems to be quite diverse and dynamic. It includes a mix of commercial, residential, and green or recreational areas. The route also features a variety of road types, such as primary roads, residential roads, and links to/from primary roads. The speed limits vary from 29.50 kph to 50.00 kph, and the number of lanes changes from 1 to 3. The journey is relatively short, at 4.11 km, an d should take around 5.22 minutes. Overall, it seems like a vibrant and interesting route that offers a glimpse into the different facets of the area.

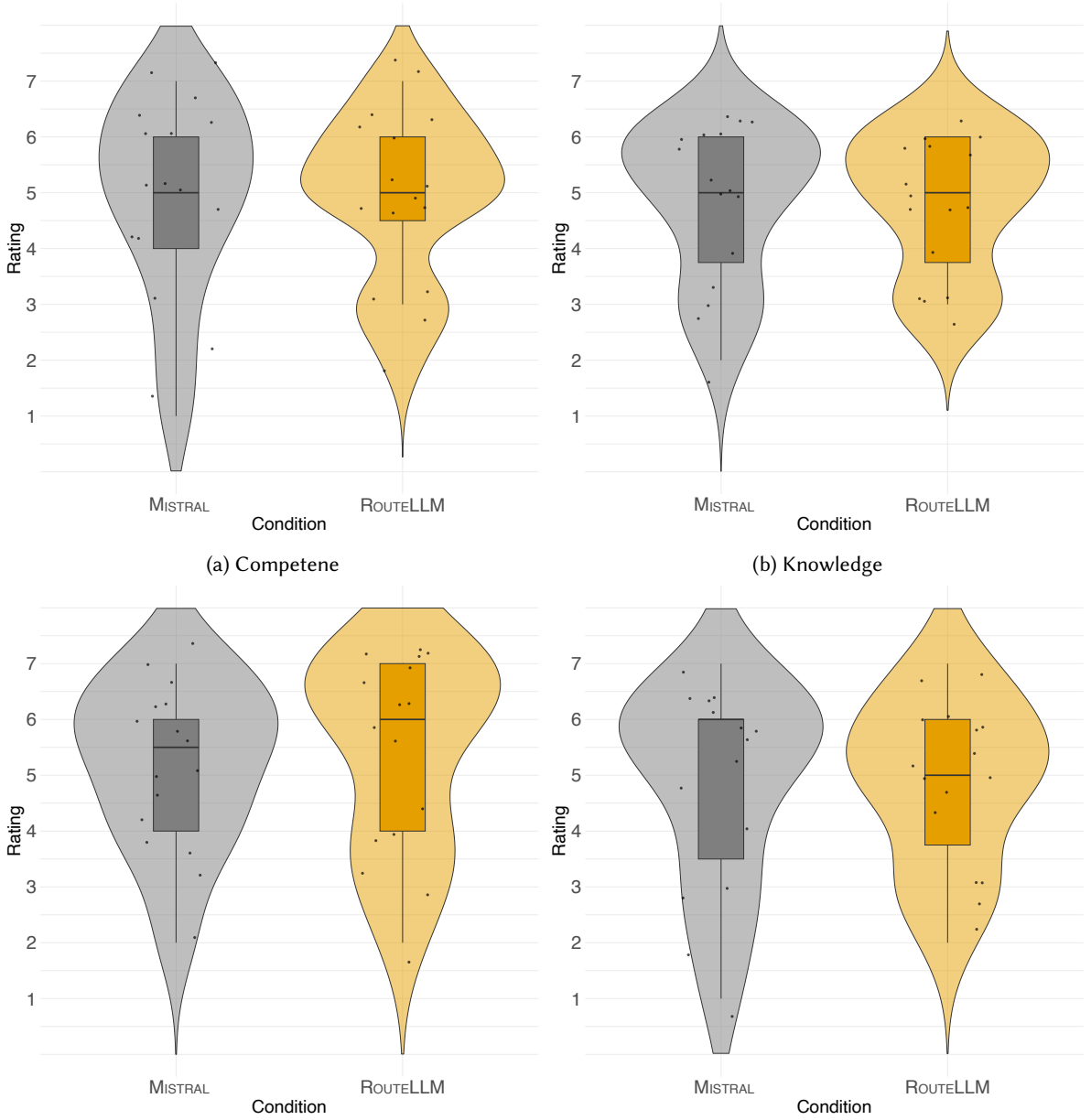
Human: How's the weather today?
RouteLLM: Great question! Today's weather seems to be quite pleasant for a journey. Throughout your trip, you can expect clear skies with a temperature of around 11.1 degrees Celsius. There won't be any rain, and the wind will be blowing at a moderate speed of 6.0 kph, with gusts reaching 12.0 kph. Enjoy your journey and make the most of the good weather!

Human: I am scared of loud sounds. Is this route suitable?
RouteLLM: The route you've chosen seems suitable for someone who is scared of loud sounds. Most of the journey is through residential areas and green or recreational areas, which should provide a more tranquil atmosphere. The speed limits are relatively low, so traffic noise should also be minimal. The road is mostly flat with gentle curves, making for a smooth drive. The weather conditions are also favorable, with no rain and mostly cloud skies, which should contribute to a comfortable journey.

Fig. 9. Examplary Conversations with *RouteLLM*.

structure understanding, the `length task` and the `long description task` require the model to extract and report exact feature values from the route, explaining the comparatively low score.

6.2.3 Conclusion. Overall, *RouteLLM* demonstrates good performance in feature extraction tasks, with a mean score of 7.02, although it sometimes struggles with the retrieval of exact values. It achieves moderate performance in Route Reasoning tasks, averaging 5.97, reflecting a general understanding of route-related questions. Similarly, it shows reasonable capability in Route Understanding tasks, but reveals a limitation in structural comprehension, particularly in the `reorder task`. This task's lower score indicates that while *RouteLLM* can grasp contextual route information, it faces challenges in deeper structural understanding.



(a) Competene

(b) Knowledge

(c) Responsibility

(d) Intelligence

Fig. 10. Participant ratings and corresponding statistics for the Perceived Intelligence questionnaire. We compared RouteLLM against Mistral, the foundation model it builds upon, to isolate the effect of our route tokenizer and training schema. Using only route tokens, RouteLLM achieved performance on par with Mistral that is provided with a full step-by-step textual description of the route. Hence, the differences across dimensions were not statistically significant.

6.3 Comparative User Study

6.3.1 Apparatus. We evaluate the usefulness of conversations with RouteLLM and its perceived intelligence in a comparative in-the-wild user study. Thereby, participants evaluated conversations with RouteLLM and another baseline system about a route. Participants did not know which system they were interacting with at which time. The following *Systems* were part of the study:

- First, RouteLLM, the first LLM with native, i.e., token-level, route-context understanding developed in this work.
- And second Mistral, the LLM used as base model for *RouteLLM*. In the study, we encode the route context as text and feed it to the network as additional input besides the user prompt.

We decided to use Mistral as the baseline model for this study, as it is the foundation model RouteLLM builds upon. Hence, comparing RouteLLM against it allows us to isolate the effect our proposed route tokenizer and training schema have on the model's performance. To input a route into Mistral, we first merge similar segments of the route and summarize each chunk using a template sentence. Thereby, we identify breakpoints by threshold or category changes, and thus compute aggregate statistics, e.g., mean or sum, per chunk to populate the template sentence. The system prompt together with and exemplary route description are shown in the Appendix A.4.

As previously explained, this form of encoding of routes is inefficient. Thus, we limit the routes' lengths to 256 tokens in this study, so that the textually encoded routes still fit the context length of Mistral. Compared to RouteLLM, Mistral does have an advantage in this study, as the task of answering questions about the route comes down to finding the right text in the prompt and generating an answer from it. Still, we compare RouteLLM against this baseline, as it can serve as an upper bound for its performance. If our concept of enabling route-context awareness through route tokens works well, RouteLLM can extract the necessary information to answer user queries from it, comparable to extracting the information from the prompt as Mistral does, and thus, RouteLLM should be able to achieve comparable performance to Mistral.

6.3.2 Procedure and Design. Before starting the study, participants gave their informed consent. Next, we collected basic demographic data, such as self-identified gender, age, familiarity with-, usage of-, and attitude toward generative AI systems. At the beginning of the study session, participants were explained that the goal of the study was to explore the route context understanding of the systems and the usefulness of their responses. They were not told how the systems work. Next, we asked the participants to give the start- and end-point of a route they are very familiar with, for example, their way to work. This was to ensure that they could evaluate the factual accuracy and usefulness of the answers given by the systems to queries about the route. We showed the participants two examples of potential questions they could ask, i.e., questions that require route context understanding, and two examples of questions that might not be asked, i.e., questions that require retrieval of non-contextual information. The examples are shown in the Appendix A.5. Then, we started the first trial, where participants interacted freely for 8 minutes with the first system. Immediately after the interaction, participants filled out a questionnaire asking them about the perceived intelligence of the system [6]. Further, they were asked to rate the answers given by the system in terms of their factual accuracy, usefulness, and overall quality on a 7-point Likert scale. Finally, they had the chance to also give qualitative feedback as free text input. This procedure was repeated once per system. After each trial, participants were asked if they needed a short break. After finishing the questionnaire for the second system, participants were asked which of the two systems they would prefer to use. The whole session took approximately 30 minutes.

The experiment was a one-factor within-subjects design. The independent variable was the *System* with two levels RouteLLM and Mistral. *System* was counter-balanced between subjects using a Latin square to account for order effects.

6.3.3 Participants. Sixteen volunteers were recruited via a call for participation, eight of whom self-identified as male, eight self-identified as female, and none wished not to answer. Their ages ranged from 21 to 30 years (*mean*=26.50, *SD*=3.20). None of the participants had any prior experience with the system before the study. All participants reported to be familiar with generative AI systems (8 Agree, 8 Strongly Agree) and the majority stated to be frequent users of them (1 Disagree, 1 Slightly Agree, 5 Agree, and 9 Strongly Agree). Further, most participants agreed that AI systems in the context of car rides are important (1 Disagree, 1 Slightly Disagree, 1 Neutral, 3 Slightly Agree, 9 Agree, 1 Strongly Agree).

6.3.4 Quantitative Results. The measures of all 7-point Likert results were analyzed using Friedman's test. Posthoc comparison was conducted using a Wilcoxon Signed-Rank test with Holm's sequential Bonferroni procedure [25]. We found RouteLLM to perform on par with Mistral across all four dimensions of the perceived intelligence questionnaire, i.e. there was no significant effect of *System* on Competence (Mistral-*Mean*: 4.94 vs RouteLLM-*Mean*: 4.94, W = 42.5, p = .86), Knowledge (Mistral-*Mean*: 4.81 vs RouteLLM-*Mean*: 4.81, W = 38.0, p = .97), Responsibility (Mistral-*Mean*: 5.19 vs RouteLLM-*Mean*: 5.38, W = 35.0, p = .78), and Intelligence (Mistral-*Mean*: 4.80 vs RouteLLM-*Mean*: 4.88, W = 31.0, p = .89). We visualize an overview of the results of the perceived intelligence questionnaire in Figure 10. Additionally, participants ratings indicate that RouteLLM's performance was on par with that of Mistral as there was no significant effect of *System* on the Factual Accuracy (Mistral-*Mean*: 4.75 vs RouteLLM-*Mean*: 5.56, W = 20.5, p = .15), the Usefulness (Mistral-*Mean*: 4.38 vs RouteLLM-*Mean*: 4.75, W = 29.0, p = .45), and the Overall Quality (Mistral-*Mean*: 5.19 vs RouteLLM-*Mean*: 5.12, W = 38.5, p = 1.0) of the answers. Finally, out of 16 participants, nine stated that they preferred RouteLLM over Mistral, six preferred Mistral, and one couldn't tell.

6.3.5 Qualitative Results. We conducted a thematic analysis following the approach outlined by Braun et al. [14] for the qualitative feedback the participants gave on RouteLLM. In a team consisting of two coders, both researchers conducted open coding of the data individually. Finally, the codes of both researchers were grouped and the following four key themes established: *Information Quality*, *Safety Boundaries*, *Conversational Behavior*, and *Feature Requests*.

- **Information Quality**: Participants appreciated the depth and richness of the system's responses. Descriptions ranged from "*Very detailed*" [P2] and "*detailed and very thorough*" [P12] to "*great additional information*" (P14), emphasizing the perceived comprehensiveness of the answers. Further, the users determined the answers generated by RouteLLM to be mostly correct. They described the answers as "*precise*" [P14], "*factually correct*" and with "*truthful facts*" [P4].
- **Safety Boundaries**: Participants highlighted the system's cautious behavior, especially regarding safety-related queries. Statements like "*very cautious*" [P1], "*focused on safety*" [P2] reflected a positive perception, while others perceived it as limiting the usefulness: "*Not helpful with questions about security*" [P10].
- **Conversational Behavior**: The system's style was perceived as professional and pleasant ("*Language was casual and pleasant*" [P16]; "*eloquent*" [P4]), but sometimes repetitive or unclear ("*it repeated 'That's a good question' multiple times*" [P10]; "*Answers in better format needed*" [P6]).
- **Feature Requests**: Some participants expressed a desire for additional or specific information, as reflected by P8: "*In some cases, information was missing*".

6.3.6 Summary. Thus, overall, the results show that RouteLLM can understand information encoded in the route tokens and provide its users with knowledgeable answers at the same level users are used to from interacting with LLMs. In order to encode the route context in a textual format for the Mistral baseline, we had to limit the route lengths to 256 tokens. A key advantage of RouteLLM for in-the-wild applications is that it can support much longer routes as its token-level route context understanding allows to input routes in a much more compact and scalable representation. Moreover, both the quantitative and qualitative results highlight that users overall

perceive RouteLLM's answers as factually accurate, helpful, and professionally phrased, further confirming the model's practical applicability for real-world route-based reasoning tasks.

In this study, we chose Mistral as the baseline model because it is the foundation upon which RouteLLM is built. This design choice enables a controlled comparison that isolates the effect of our proposed route tokenizer and training schema on model performance. Importantly, the core concept of RouteLLM, introducing a dedicated route tokenizer for structured context representation, is agnostic to the underlying foundation model. Therefore, it could be integrated with more recent language models such as DeepSeek. RouteLLM performing on par with Mistral in our study, even though Mistral was prompted with detailed, step-by-step textual route descriptions, shows that LLMs can be used to understand route context with a dedicated route tokenizer. While we expect our methodology to generalize to larger and more capable foundation models, an evaluation of this should be part of future work.

7 DISCUSSION

7.1 Failure Cases

Overall, we can report that *RouteLLM* is capable of understanding route context and zero-shot reasoning based on the world knowledge encoded during pretraining. Still, it occasionally suffers from inaccuracies, hallucinations, fluency degenerations, or unrelated responses. Thus, we find that further advances in cross-modal understanding between route-context and text will become necessary. During our evaluations, we found the following most prominent failure cases of *RouteLLM*:

- **Feature Value Inaccuracy:** Some of the model generations contain inaccurate route context feature values. The level of inaccuracy ranges from minor errors, such as underestimating the temperature by a few degrees, to major errors, such as giving a route length that is multiple kilometers longer.
- **Hallucinations:** Occasionally, the model answers the original question but then follows with any unnecessary or false information. For example, the model correctly responds that there are bodies of water along the route but then includes recreational and commercial areas, which the route does not pass through.
- **Fluency Degeneration:** Generally, the model has a high fluency, and the generated responses are coherent. However, the model's answers occasionally suffer from a degeneration of language fluency. In those cases, the responses can be wrongly formatted, have wrong or confusing wording, or sound unnatural. We also find that the model tends to start responses with phrases such as *Great!* or *Great question!* or *I'm glad you found these suggestions helpful!* even if they do not relate to the previously asked question.
- **Unrelated Responses:** A few model responses were not related to the user prompt after all. This includes the model ignoring parts of the user prompt, the model answering a different route-related question, or the model generating a coherent but unrelated response. For example, the model repeatedly generated the answer *Great choices! These ideas will surely make for an unforgettable 10-year-old's birthday celebration.* in our evaluation. The mentioned responses occur mainly if the input routes are longer than 256 tokens or if the user prompt contains questions the model cannot answer.

7.2 Challenges

RouteLLM demonstrated emerging abilities to reason based on route context. While our analysis also revealed clear limitations, we see it as a first step towards- and future benchmark for route context-aware systems with reasoning capabilities. Based on the evaluation results and the identified failure cases, we identify three key challenges for future research on route-context-aware LLMs:

Challenge 1: Routes are a Unique Modality that Requires Special Attention. Our proposed *Route Tokenizer* and *RouteLLM* are heavily inspired by MotionGPT [28], a model that demonstrated cross-modal understanding

between human motion and natural language. However, there are key differences between human motion and routes, leading to the approach being not fully transferable to routes. When expressing a human motion in text, no underlying feature values must be exactly repeated. The numerical values of the joint positions and rotations are irrelevant to the LLM, which instead focuses on the general motion. On the other, the LLM must be capable of extracting the exact underlying feature values for routes. While there are abstract concepts that the model can gather from the feature values, they are not sufficient to perform most of the interactions a user would request, such as calculating the length of a route. The reliance on the exact feature values makes the LLM sensitive to variations of the input features, where a small error in the route tokenizer results in a high error in the LLM outputs or even an utterly incorrect response. We believe that input sensitivity, combined with the need for exact feature values and lossy route tokens, is the primary reason for incorrect feature values in *RouteLLM's* generations.

Thus, a primary challenge lies in the inherent sensitivity of route data. Routes are composed of precise numerical features, such as distances, road conditions, and traffic, that require accurate encoding and decoding to maintain fidelity. Small errors during tokenization can lead to significant inaccuracies in model responses, which is problematic for user interactions that rely on exact details. This demand for precision makes cross-modal understanding of routes particularly challenging, as traditional LLMs are not designed to handle high sensitivity to input features. Overcoming this challenge requires developing robust tokenization methods that preserve the integrity of route data without introducing errors that could mislead the model's output.

Challenge 2: LLMs Lacking Mathematical Capabilities Are Problematic when Dealing with Routes. A subset of tasks on numerical features requires the model to be capable of mathematical reasoning, e.g., calculating the total length of a route or finding the maximum temperature along a route. Compared to the base model used in this work (Mistral 7B), newer, larger models perform significantly better on mathematical benchmarks such as MATH [23] and GSM-8K [16]. Still, these models' mathematical abilities are limited, which might lead to inaccurate or incomplete responses on route-context based tasks. This inability potentially limits the models' practical utility when dealing with contextual routes. Improving the mathematical reasoning of LLMs is crucial to ensuring that they can reliably process and interpret numerical route features, thus enabling more effective and accurate user interactions.

Challenge 3: Adding Additional Context Features Is Challenging. The architecture of *RouteLLM* relies on tokenization for both text and route context. For natural language, a Byte Pair Encoding (BPE) tokenizer converts text into text tokens, provided the symbols exist in the vocabulary. Similarly, our newly introduced route tokenizer converts route context features into route tokens. However, this design presents a challenge when additional context features must be incorporated. Since the existing route tokenizer is trained on a fixed set of features, it cannot directly accommodate new ones, making expansion difficult. This challenge is comparable to adding support for a new language with entirely different characters or symbols to an LLM. A possible solution could consist in the introduction of an additional tokenizer or the replacement of the current tokenizer with one that supports a broader set of context features. Another approach might be to represent additional context features through the ones supported by the tokenizer, comparable to the concept of "paraphrasing", where a missing word or concept is expressed through alternative phrasing. Still, further research is needed to explore more sophisticated methods for extending route context-aware LLMs' support to additional context features.

8 CONCLUSION

In this work, we took a first step in the direction of large language models (LLMs) that can understand and reason with route context data. We introduced two novel datasets, *ContextualRoutes* and *LabeledRoutes*, which contain over 300k routes and labels for 1000 tasks, allowing to fine-tune pretrained LLMs for route-context-based

reasoning. We proposed a new *Route Tokenizer* that converts route context into tokens analogous to natural language tokens and developed *RouteLLM*, the first LLM capable of understanding route context natively, i.e., at the token-level. *RouteLLM* allows users to ask complex questions about their routes in natural language that require world knowledge and semantic understanding, such as determining the scenic value or estimating the best time for a phone call. We demonstrated that after fine-tuning, *RouteLLM* can even perform zero-shot reasoning on route context data, i.e. give meaningful answers to questions unseen during training. While our system demonstrates a promising glimpse into the future, we identified several key challenges for future research. We hope, that our work contributes to the growing body of work aimed at integrating LLMs into practical, context-aware applications in dynamic environments.

References

[1] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. *arXiv preprint arXiv:1906.02569* (2019).

[2] Utku Günay Acer, Marc van den Broeck, Chulhong Min, Mallesham Dasari, and Fahim Kawsar. 2022. The city as a personal assistant: turning urban landmarks into conversational agents for serving hyper local information. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–31.

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

[4] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 929–947. doi:10.1145/3620665.3640366

[5] Apple Inc. 2024. Apple Intelligence Comes to iPhone, iPad and Mac starting next month. https://www.apple.com/de/newsroom/2024/09/apple-intelligence-comes-to-iphone-ipad-and-mac-starting-next-month/. [Accessed 20-09-2024].

[6] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. 2009. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International journal of social robotics* 1 (2009), 71–81.

[7] Jan Henry Belz, Lina Madlin Weilke, Anton Winter, Philipp Hallgarten, Enrico Rukzio, and Tobias Grosse-Puppendahl. 2024. Story-Driven: Exploring the Impact of Providing Real-time Context Information on Automated Storytelling. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–15.

[8] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432* (2013).

[9] David Bethge, Luis Falconeri Coelho, Thomas Kosch, Satiyabooshan Murugaboopathy, Ulrich von Zadow, Albrecht Schmidt, and Tobias Grosse-Puppendahl. 2023. Technical Design Space Analysis for Unobtrusive Driver Emotion Assessment Using Multi-Domain Context. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–30.

[10] David Bethge, Constantin Patsch, Philipp Hallgarten, and Thomas Kosch. 2023. Interpretable Time-dependent convolutional emotion recognition with contextual data streams. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–9.

[11] Prabin Bhandari, Antonios Anastasopoulos, and Dieter Pfoser. 2023. Are Large Language Models Geospatially Knowledgeable?. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems* (Hamburg, Germany) *(SIGSPATIAL '23)*. Association for Computing Machinery, New York, NY, USA, Article 75, 4 pages. doi:10.1145/3589132.3625625

[12] Temistocle Birsan and Dan Tiba. 2006. One hundred years since the introduction of the set distance by Dimitrie Pompeiu. In *System Modeling and Optimization: Proceedings of the 22nd IFIP TC7 Conference held from July 18–22, 2005, in Turin, Italy 22*. Springer, 35–39.

[13] Geoff Boeing. 2024. Modeling and Analyzing Urban Networks and Amenities with OSMnx. (2024).

[14] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[15] Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. 2024. Driving with LLMs: Fusing Object-Level Vector Modality for Explainable Autonomous Driving. *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024), 14093–14100.

[16] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

[17] Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*.

[18] Soham Deshmukh, Benjamin Elizalde, Rita Singh, and Huaming Wang. 2023. Pengi: An Audio Language Model for Audio Tasks. *Advances in Neural Information Processing Systems* 36 (2023), 18090–18108.

[19] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems* 36 (2024).

[20] Anna-Katharina Frison, Philipp Wintersberger, Tianjia Liu, and Andreas Riener. 2019. Why do you like to drive automated? a context-dependent analysis of highly automated driving to elaborate requirements for intelligent user interfaces. In *Proceedings of the 24th international conference on intelligent user interfaces*. 528–537.

[21] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. 2022. Generating Diverse and Natural 3D Human Motions from Text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5152–5161.

[22] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

[23] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving with the MATH Dataset. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=7Bywt2mQsCe

[24] John Hewitt. 2021. Initializing New Word Embeddings for Pretrained Language Models. https:/nlp.stanford.edu/~johnhew//vocab-expansion.html.

[25] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[26] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[27] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[28] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. 2024. MotionGPT: Human Motion as a Foreign Language. *Advances in Neural Information Processing Systems* 36 (2024).

[29] Mohamed Kari, Tobias Grosse-Puppendahl, Alexander Jagaciak, David Bethge, Reinhard Schütte, and Christian Holz. 2021. Soundsride: Affordance-Synchronized Music Mixing for In-Car Audio Augmented Reality. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 118–133.

[30] Eryk Lewinson. 2022. Three Approaches to Encoding Time Information as Features for ML Models. NVIDIA Developer Blog. Accessed 2024-09-02.

[31] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network That Explains Its Predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[32] Zekun Li, Wenxuan Zhou, Yao-Yi Chiang, and Muhao Chen. 2023. GeoLM: Empowering Language Models for Geospatially Grounded Language Understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 5227–5240. doi:10.18653/v1/2023.emnlp-main.317

[33] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved Baselines with Visual Instruction Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 26296–26306.

[34] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual Instruction Tuning. *Advances in Neural Information Processing Systems* 36 (2024).

[35] Rohin Manvi, Samar Khanna, Gengchen Mai, Marshall Burke, David B. Lobell, and Stefano Ermon. 2024. GeoLLM: Extracting Geospatial Knowledge from Large Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=TqL2xBwXP3

[36] Bruce Mehler, Bryan Reimer, Lisa A D'Ambrosio, Alexander Piña, and Joseph F Coughlin. 2010. An Evaluation of Time of Day Influences on Simulated Driving Performance and Physiological Arousal. In *Proceedings of the 89th Annual Meeting of the Transportation Research Board on Traffic and Transport Planning*. 1–15.

[37] Peter Mooney, Wencong Cui, Boyuan Guan, and Levente Juhász. 2023. Towards Understanding the Geospatial Skills of ChatGPT: Taking a Geographic Information Systems (GIS) Exam. In *Proceedings of the 6th ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery* (Hamburg, Germany) *(GeoAI '23)*. Association for Computing Machinery, New York, NY, USA, 85–94. doi:10.1145/3615886.3627745

[38] Matthias Plappert, Christian Mandery, and Tamim Asfour. 2016. The KIT Motion-Language Dataset. *Big Data* 4, 4 (dec 2016), 236–252. doi:10.1089/big.2016.0028

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning Transferable Visual Models from Natural Language Supervision. *International Conference on*

Machine Learning, 8748–8763.
[40] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems* 32 (2019).
[41] Jonathan Roberts, Timo Lüddecke, Sowmen Das, Kai Han, and Samuel Albanie. 2023. GPT4GEO: How a Language Model Sees the World's Geography. *arXiv preprint arXiv:2306.00020* (2023).
[42] R Tyrrell Rockafellar and Roger J-B Wets. 2009. *Variational analysis*. Vol. 317. Springer Science & Business Media.
[43] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: an adversarial winograd schema challenge at scale. *Commun. ACM* 64, 9 (Aug. 2021), 99–106. doi:10.1145/3474381
[44] Rob Semmens, Nikolas Martelaro, Pushyami Kaveti, Simon Stent, and Wendy Ju. 2019. Is Now A Good Time? An Empirical Study of Vehicle-Driver Communication Timing. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.
[45] Smitha Sheshadri and Kotaro Hara. 2024. Conversational Localization: Indoor Human Localization through Intelligent Conversation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 4 (2024), 1–32.
[46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.
[47] Arash Tavakoli, Vahid Balali, and Arsalan Heydarian. 2020. A Multimodal Approach for Monitoring Driving Behavior and Emotions. Mineta Transportation Institute.
[48] Arash Tavakoli, Mehdi Boukhechba, and Arsalan Heydarian. 2020. Personalized Driver State Profiles: A Naturalistic Data-Driven Study. In *Proceedings of the AHFE 2020 Virtual Conference on Human Aspects of Transportation*. Springer, 32–39.
[49] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems* 30 (2017).
[50] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).
[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.
[52] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6
[53] Tong Wu, Nikolas Martelaro, Simon Stent, Jorge Ortiz, and Wendy Ju. 2021. Learning When Agents Can Talk to Drivers Using the INAGT Dataset and Multisensor Fusion. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–28.
[54] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2023. A Survey on Multimodal Large Language Models. *arXiv preprint arXiv:2306.13549* (2023).
[55] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 4791–4800.
[56] Hang Zhang, Xin Li, and Lidong Bing. 2023. Video-LLaMA: An Instruction-Tuned Audio-Visual Language Model for Video Understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Yansong Feng and Els Lefever (Eds.). Association for Computational Linguistics, Singapore, 543–553. doi:10.18653/v1/2023.emnlp-demo.49
[57] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems* 36 (2024).

A APPENDIX

A.1 Bounding Box Locations for *ContextualRoutes*

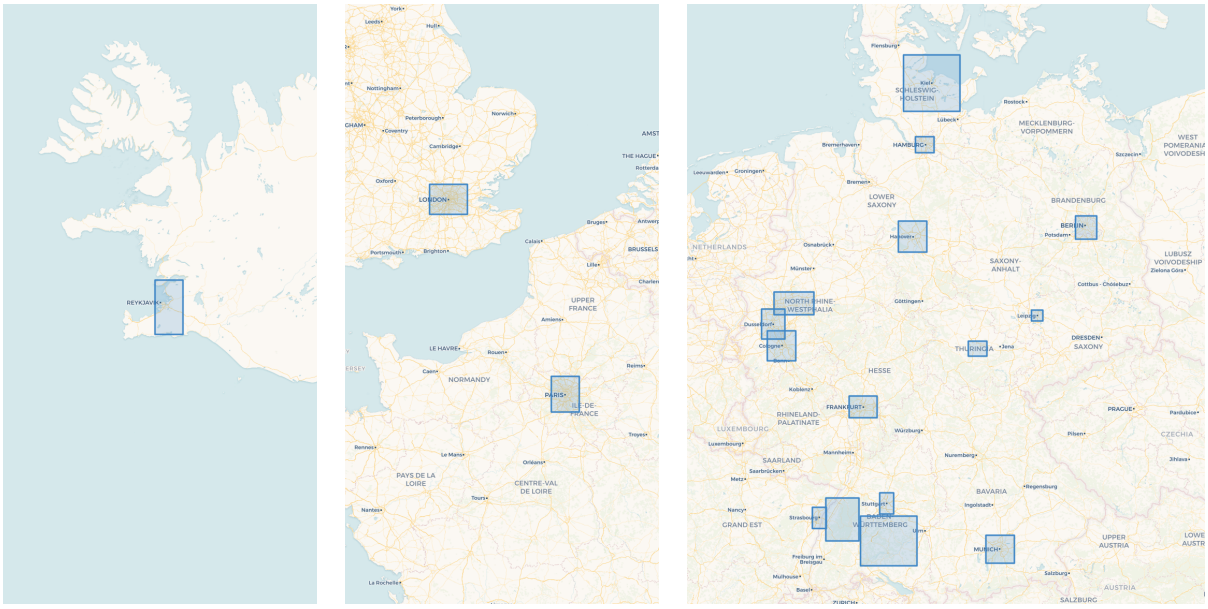


Fig. 11. We sample routes from 18 bounding boxes in order to create the *Contextual Routes* and *LabeledRoutes* datasets. The routes comprise a variety of route context settings, such as rural areas or urban areas.

Table 2. Bounding Box Locations

Split	City	Approximate Area (km^2)	Center Coordinate
Training	Berlin	2700	(52.4935, 13.4066)
	Düsseldorf	3900	(51.23, 6.7733)
	Erfurt	1600	(50.8968, 11.106)
	Essen and Dortmund	5000	(51.4997, 7.2123)
	Frankfurt	3500	(50.1143, 8.6785)
	Hamburg	1700	(53.5486, 9.9875)
	Hannover	4900	(52.3770, 9.7279)
	Kiel	17800	(54.3175, 10.1355)
	Munich	4400	(48.1348, 11.5820)
	Stuttgart	1700	(48.7831, 9.1815)
	Tübingen	15600	(48.2546, 9.2220)
	Paris	5600	(48.8580, 2.35)
	Reykjavik	8500	(64.1018, −21.7621)
	Strassburg	1700	(48.5795, 7.7496)
Test	Black Forest	7800	(48.5560, 8.2384)
	Cologne	4700	(50.9392, 6.952)
	Leipzig	740	(51.3408, 12.3709)
	London	6400	(51.5078, −0.1257)

A.2 Overview over the *ContextualRoutes* and *LabeledRoutes* Datasets

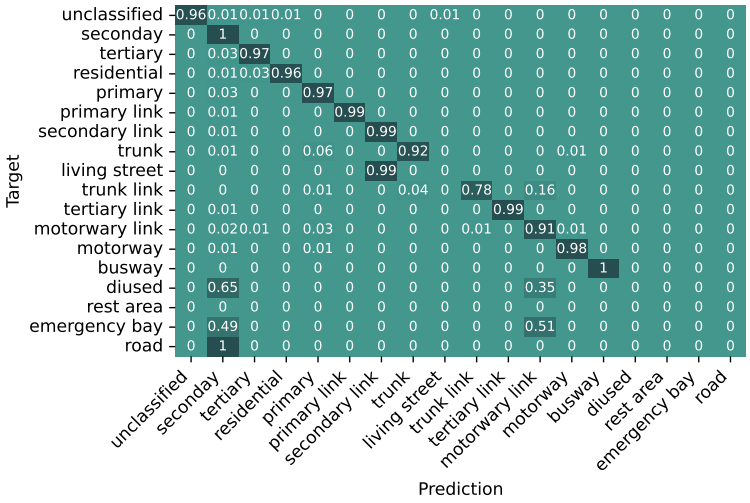
Table 3. We present an overview of all context features used to describe the route segments in the *ContextualRoutes* and *LabeledRoutes* datasets.

	Feature Name	Source / Origin	Example	Description
Temporal	cos_time	System	0.5	Cosine Component of the local Time Measurement Encoding [30]
	sin_time	System	0.5	Sine Component of the local Time Measurement Encoding [30]
	weekday	System	True	Flag for weekdays
Weather	temperature	Azure Maps API	22.5 [°C]	Temperature
	cloud_cover	Azure Maps API	60 [%]	Percentage of sky that is obscured by clouds
	precipitation	Azure Maps API	10 [dBZ]	Precipitation in dBZ
	wind_speed	Azure Maps API	15 [kph]	Speed of the wind
	wind_gust_speed	Azure Maps API	25 [kph]	Speed of the wind gusts
	wind_direction	Azure Maps API	180 [°]	Heading of the winds
	lightning	Azure Maps API	False	Flag for lightning
Road	lanes	OSMnx graph	2	Number of lanes
	length	OSMnx graph	150 [m]	Length of the segment
	heading	OSMnx graph	90 [°]	Heading of the segment
	curvature	OSMnx graph	0.05 [m^{-1}]	Curvature of the segment
	highway	OSMnx graph	*residential*	Type of road
	oneway	OSMnx graph	True	Flag for oneway streets
	tunnel	OSMnx graph	False	Flag for tunnels
	bridge	OSMnx graph	True	Flag for bridges
	has_junction	OSMnx graph	False	Flag for junctions
	reversed	OSMnx graph	False	Flag for segment reversal (OSM node order)
	landuse	OSMnx graph	*commercial*	Describes what the area next to the segment is used for
	landuse_distance	OSMnx graph	50 [m]	Distance of the area to the edge
	grade	JAXA	-0.03	Grade of the segment
	grade_abs	JAXA	0.03	Absolute grade of the segment
Traffic	speed_kph	OSMnx graph	50 [kph]	The speed limit
	travel_time	OSMnx graph	120 [s]	OSM estimated travel time of the segment
	max_speed_variable	OSMnx graph	False	Flag for roads with variable speed limit
	unlimited_speed	OSMnx graph	False	Flag for roads without a speed limit
	free_flow_speed	Azure Maps API	55 [kph]	Azure Maps estimated free flow speed of the segment
	delay	Azure Maps API	0.8	Delay due to traffic compared to free flow speed
	current_speed	Azure Maps API	45 [kph]	Current traffic speed
	current_travel_time	Azure Maps API	140 [s]	Current segment travel time
	flow_road_closure	Azure Maps API	False	Indicates whether the road is closed at the segment
	incident_category	Azure Maps API	*Accident*	Category of the incident
	incident_delay	Azure Maps API	30 [s]	Delay caused by the incident in seconds
	incident_distance	Azure Maps API	100 [m]	Distance of the incident from the segment
	incident_certainty	Azure Maps API	0.9	Certainty of the incident
	incident_magnitude	Azure Maps API	2	Impact of the incident
	incident_reported	Azure Maps API	True	Flag for reported incidents

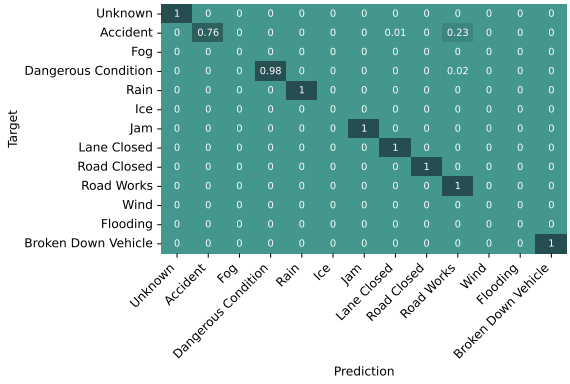
Table 4. We annotate the routes in the *ContextualRoutes* dataset with labels that allow to train *RouteLLM* using various tasks. We present an overview of these tasks in this table. A † indicates that numerical values were converted to natural language categories.

	Task Name	Annotation Strategy	Description
Feature Extraction	bridge	Template & Teacher	Whether there is a bridge along the route.
	flow-road-closure	Template & Teacher	Whether there is a road closure along the route.
	has-junction	Template & Teacher	Whether there is a junction along the route.
	incident-reported	Template & Teacher	Whether there is an incident along the route.
	lightning	Template & Teacher	Whether there is lightning along the route.
	max-speed-variable	Template & Teacher	Whether there is a variable max speed along the route.
	oneway	Template & Teacher	Whether there is a oneway street along the route.
	tunnel	Template & Teacher	Whether there is a tunnel along the route.
	unlimited speed	Template & Teacher	Whether the speed is unlimited along the route.
	weekday	Template & Teacher	Whether it is a weekday.
	cloud-cover	Template & Teacher	Amount of cloud cover along the route†.
	road-type	Template & Teacher	Questions on the category of road type along the route.
	incident category	Template & Teacher	The category of any incident along the route.
	incident certainty	Template & Teacher	Certainty for reported incidents along the route†.
	incident magnitude	Template & Teacher	The magnitude of any incident along the route†.
	landuse	Template & Teacher	Type of land use along the route (e.g., urban, rural).
	lanes	Template & Teacher	Number of lanes along the route.
	precipitation	Template & Teacher	Amount of precipitation along the route†.
	length	Template & Teacher	The total length of the route.
Route Understanding	compare-feature	Template	Compare two routes based on a feature condition.
	feature-sequence	Template	Longest subsequence meeting a specified feature condition.
	continuation	Template	Find the correct continuation of a route in a list of three options (inspired by [55]).
	masking	Template	Predict a masked route token.
	reorder	Template	Put 3 to 26 route parts in the correct chronological order.
	localization	Template	Locate a subsequence at the route's beginning, middle, or end.
	sorting	Template	Order routes by a specified feature (ascending or descending).
Route Reasoning	beauty	Teacher	Answer questions on route beauty, considering structure, weather, and traffic.
	creative writing	Teacher	Write a dialog based on a scenario and a route.
	diverse questions	Teacher	Answer creative questions about the route, like writing poems or suggesting songs.
	quiz	Teacher	Create a quiz on the route and provide the correct answers.
	landscape	Teacher	Describe the landscape surrounding the route.
	long description	Teacher	Aa description of the route of at least ten sentences.
	sentence description	Teacher	Describe the route in one sentence.
	short description	Teacher	A short description of the route with at max five sentences.
	traffic report	Teacher	Answer questions about route traffic, including reporting, identifying dangers, and assessing comfort.
	weather report	Teacher	Answer questions about route weather, including reporting, identifying obstacles, and assessing weather implications.

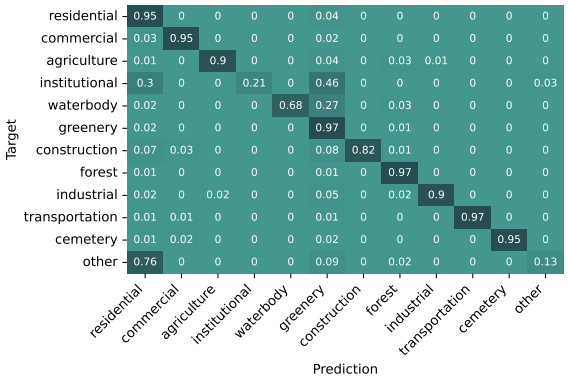
A.3 Confusion Matrices of the VQ-VAE Reconstructions



(a) `road_type`



(b) `incident_category`



(c) `landuse`

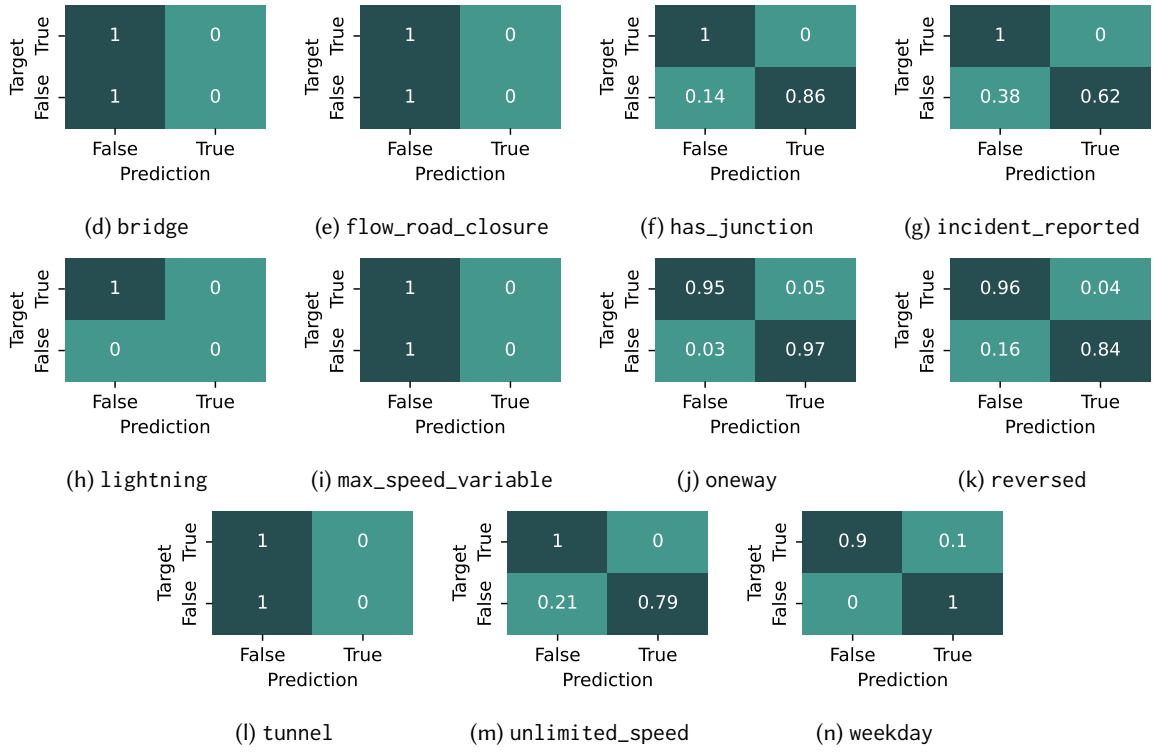


Fig. 12. Confusion Matrices of the VQ-VAE Reconstructions for the Categorical Features (a) - (c), and the Binary Features (d) - (n).

A.4 Prompt Used for Mistral in the User Study

```
You are an AI that answers questions about routes. Take a deep breath and
do it step by step. It is crucial that you are very concise and give short
answers. Step ids are marked as id#$step_number.

Reply regarding the following route and do not mention any step ids in
your text: The route is described by the following segments.

The given route consists of 27 steps. The first 11 steps are on a
residential road with an average grade of -2%, an average curvature of
0.0025, and an average distance of 0 meters from the street to industrial
areas. The free flow speed is 44.64 kph, and the current average traffic
speed is 45.01 kph.
```

```
Steps 12 to 16 have an average grade of -1%, an average curvature of
0.00905, and an average distance of 9 meters from the street to
residential areas. The free flow speed is 30.00 kph, and the current
average traffic speed is 30.00 kph.

Step 17 has an average grade of 2%, a straight average curvature, and an
average distance of 5 meters from the street to transportation areas. The
free flow speed is 30.00 kph, and the current average traffic speed is
30.00 kph.

Steps 18 to 21 have an average grade of -4%, an average curvature of
0.0002, and an average distance of 9 meters from the street to residential
areas. The free flow speed is 30.00 kph, and the current average traffic
speed is 30.00 kph.

Steps 22 to 25 have an average grade of flat, an average curvature of
0.00873, and an average distance of 9 meters from the street to commercial
areas. The free flow speed is 30.00 kph, and the current average traffic
speed is 30.00 kph.

Steps 26 and 27 have an average grade of 5%, an average curvature of
0.03003, and an average distance of 10 meters from the street to
commercial areas. The free flow speed is 30.00 kph, and the current
average traffic speed is 22.88 kph.
```

A.5 Examples Used for the User Study

We showed the participants in our user study with examples for suitable (⊕) and unsuitable (⊖) questions to ask the agents:

- ⊕ How is the traffic along my route?
- ⊕ What would be the best moment for a phone call along my route?
- ⊕ Is this a scenic route?
- ⊖ Is there an open restaurant nearby my route?
- ⊖ What is the name of the street I am driving on?
- ⊖ What is the name of the city on the right?