Animal Transport Planning with LLM Agent

Bolgov Maxim

December 2025

Abstract

This project proposes a solution to the problem of animal transport planning based on minimal user input: a photograph of an animal and start/end locations. The system combines image captioning, geospatial routing, and weather data through an LLM agent to recommend suitable transport modes and provide time estimates. The agent is built using PydanticAI with Qwen 2.5-7B as the reasoning engine, leveraging function calling to orchestrate multiple external APIs. Florence-2-base handles image-to-text conversion, Yandex Geocoder resolves addresses to coordinates, OpenRouteService provides routing information, and OpenWeatherMap supplies meteorological data. The agent integrates these tools to produce weather-aware, distance-sensitive transport recommendations while supporting multi-turn dialogue for refinement. Implementation was constrained to open source models and free APIs, deployed on legacy hardware (2× T4 GPUs).

GitHub repository: https://github.com/defdet/animal_transport_agent

1. Introduction

Animal transport planning is a practical task that requires consideration of multiple factors: the physical characteristics of the animal (size, species-specific needs), the distance and duration of travel, and environmental conditions such as temperature or wind speed. While specialized logistics systems exist for batch animal transfers between shelters, and routing optimization is well-studied in general logistics, there is no publicly available system that combines multimodal input (image + text) with real-time geospatial and meteorological data to provide individualized transport recommendations.

The rise of large language models (LLMs) capable of tool use enables a new approach: an agent that can interpret natural language queries, coordinate calls to external APIs, and synthesize structured recommendations. Recent work demonstrates that LLMs augmented with external tools can overcome limitations in world knowledge and hallucination, making them suitable for real-world planning tasks. However, most agent frameworks rely on proprietary models (GPT-4, Claude) or high-end infrastructure. This work explores whether a small open-source model (7B parameters) can effectively orchestrate multiple APIs under hardware and budget constraints.

Our system addresses a well-defined use case: given an image of a single animal and two addresses within Russia, produce a recommendation for transport mode (personal car, bicycle, courier service, or air travel) along with an estimated travel time. The agent integrates vision (image captioning), geolocation (address geocoding), routing (distance/time calculation), and environmental data (weather conditions) to inform its

decision. To the best of our knowledge, this is the first implementation of an end-to-end LLM agent for individual animal transport planning using only open-source models and free APIs.

2. Related Work

2.1 Animal Logistics and Transport Systems

Wagner and Moore developed an animal transfer logistics support tool for optimizing shelter-to-shelter animal transfers. Their system consists of two components: a shelter allocator that identifies optimal partnership opportunities, and a multi-pickup delivery route scheduler built on Google OR-Tools[5]. The scheduler minimizes travel costs while respecting constraints such as vehicle capacity and pickup-before-delivery ordering. While this work demonstrates the viability of computational approaches to animal logistics, it focuses on batch optimization for professional transport operators rather than individual transport planning. Additionally, it does not incorporate image-based animal identification or weather-aware decision making.

The International Air Transport Association (IATA) publishes the Live Animals Regulations (LAR), which establish mandatory standards for air transport of live animals. The regulations specify container requirements, temperature ranges, ventilation standards, and documentation needs on a per-species basis. For example, brachycephalic dog breeds are subject to additional restrictions during hot weather due to respiratory vulnerability. While our system does not implement full LAR compliance, the regulations provide domain context for weather-based transport recommendations.

2.2 Vision Models for Animal Recognition

Recent advances in vision-language models (VLMs) have enabled zero-shot animal classification without task-specific training. Dussert et al. evaluated foundation models including GPT-4V and VideoPrism for animal behavior classification, demonstrating that VLMs can outperform vision-only specialist models in zero-shot settings[6]. This finding supports the use of general-purpose image captioning models for species identification rather than training a dedicated classifier.

Microsoft's Florence-2 is a prompt-based vision foundation model capable of multiple vision tasks including detailed captioning, object detection, and visual grounding[7]. Florence-2-base uses a DaViT vision encoder and a BART-based language decoder, trained on extensive image-text pair datasets. The model supports task-specific prompting (e.g., `<MORE_DETAILED_CAPTION>`) to control output granularity. For image captioning specifically, Florence-2 generates structured descriptions including object categories, spatial relationships, and attributes—sufficient for extracting animal species and approximate size information.

2.3 LLM Agents and Tool Use

The ReAct paradigm, introduced by Yao et al., demonstrates that interleaving reasoning traces with action execution (tool calls) enables LLMs to solve multi-step problems more reliably than chain-of-thought prompting alone[1]. In the ReAct framework, the model generates explicit reasoning steps ("Thought: I need to find the distance between these cities") followed by tool invocations ("Action: call_routing_api(origin, destination)"). This

approach achieved significant improvements on knowledge-intensive tasks (HotpotQA) and interactive decision-making benchmarks (ALFWorld, WebShop).

Qwen 2.5, developed by Alibaba Cloud, is a series of open-source language models ranging from 0.5B to 72B parameters[2]. The 7B-Instruct variant is specifically fine-tuned for instruction following and supports native function calling via a tool description schema. Function calling allows the model to generate structured JSON tool invocations when provided with function signatures and descriptions.

PydanticAI is a Python framework for building type-safe LLM agents with integrated tool support[3]. The framework uses Pydantic models to define tool schemas, automatically serializes function calls to JSON, validates tool outputs against declared types, and handles common failure modes (retries, timeouts). Its integration with vLLM enables efficient serving of open-source models with OpenAI-compatible endpoints.

2.4 Geospatial and Weather APIs

Routing optimization is a well-studied problem in logistics. Modern systems integrate real time traffic and weather data to improve delivery time estimates. A study on last-mile delivery optimization demonstrated a 20% reduction in delivery times by incorporating predictive weather and traffic analytics into route planning[4].

Yandex Maps provides a suite of geospatial APIs including a geocoding service (address → coordinates conversion) and routing APIs for various transport modes. The geocoder supports both structured addresses and natural language queries (e.g., "Moscow, Kremlin"), making it suitable for conversational interfaces. OpenRouteService (ORS) is an open-source routing engine supporting multiple profiles (driving, cycling, walking) and providing distance, duration, and elevation data. OpenWeatherMap offers a free tier for current weather and forecast data, including temperature, humidity, and precipitation for arbitrary coordinates.

3. System Architecture

3.1 Overview

Our system implements an LLM agent that coordinates four external tools to produce transport recommendations. The agent receives three inputs from the user: (1) an image file containing a photograph of an animal, (2) a starting location (Russian city or address), and (3) a destination location. It outputs a transport mode recommendation and an estimated travel time, with optional natural language justification.

The architecture consists of five components:

- Image captioning module: Florence-2-base model generates a textual description of the uploaded image
- LLM agent: Qwen 2.5-7B-Instruct orchestrates tool calls and reasoning
- Routing tool: Yandex Geocoder + OpenRouteService pipeline for distance/time calculation
- Weather tool: OpenWeatherMap API for origin/destination temperature data
- PydanticAI framework: Manages agent-tool interaction, type validation, and conversation state

The agent follows a sequential workflow: image → caption, addresses → routing data, routing data → weather check → final recommendation. Multi-turn conversation is supported, allowing users to refine requirements (e.g., "What if I leave at night instead?") without re-uploading the image.

3.2 Image Captioning with Florence-2

We use microsoft/Florence-2-base for image-to-text conversion. Florence-2 is a 0.23B parameter vision foundation model trained on diverse image-text datasets including COCO, Visual Genome, and proprietary web-scraped data.

For our application, we use the `<MORE_DETAILED_CAPTION>` task prompt, which instructs the model to generate comprehensive descriptions including object categories, quantities, colors, and spatial relationships. Example output for a dog image: "A medium-sized brown dog with pointed ears sitting on grass, appears to be a German Shepherd, adult, alert posture."

The captioning module runs inference independently before agent initialization. This design choice decouples vision processing from the LLM agent loop, reducing latency for subsequent tool calls and simplifying error handling (if captioning fails, the user receives immediate feedback). The caption is then passed as text context to the Qwen agent.

3.3 LLM Agent: Qwen 2.5-7B-Instruct

Qwen 2.5-7B-Instruct serves as the agent's reasoning engine. The model is a decoder-only transformer with 7.07 billion parameters, trained on multilingual data including Russian text corpora. Key capabilities relevant to our service include native function calling via JSON schema, instruction following fine-tuned on chat templates, and structured generation via prompt engineering.

Inference setup: The model is served using vLLM v0.8.5 on 2× NVIDIA T4 GPUs (16GB VRAM each). vLLM provides paged attention for efficient KV cache management, continuous batching for dynamic request batching, and an OpenAI-compatible API endpoint. We use tensor parallelism (TP=2) to distribute the model across both GPUs, with temperature=0.1 for more deterministic outputs.

Why Qwen 2.5-7B instead of Qwen 2.5-7B-VL? The vision-language variant (VL) integrates a vision encoder directly into the LLM, eliminating the need for separate captioning. However, Qwen 2.5-VL models exhibit degraded function calling reliability compared to text-only Instruct models. Since accurate tool invocation is critical for our multi-step workflow, we prioritize the text-only Instruct model's tool-use capability.

Hardware constraints: Older T4 GPUs limit vLLM version compatibility (v0.8.5 is the latest version supporting T4 hardware). This constrains model selection to Qwen 2.5-7B, chosen as a model fully supported by this vLLM version with reliable function calling.

3.4 Routing Tool (Yandex Geocoder + OpenRouteService)

The routing tool is implemented as a composite pipeline that converts user-provided locations into a travel time and distance estimate. First, the system performs forward geocoding using the Yandex Geocoder API to transform a natural-language address into WGS84 coordinates (latitude/longitude). For example, the query "Москва, Кремль" is resolved into a coordinate pair such as (55.7520, 37.6175). Yandex's geocoder supports fuzzy

matching and multiple address formats (cities, landmarks, full street addresses), which allows conversational inputs without strict formatting constraints.

Second, the tool queries OpenRouteService (ORS) to compute the route distance and duration using the directions endpoint with the driving-car profile. For example, given origin (55.7520, 37.6175) and destination (59.9343, 30.3351), ORS returns a distance approximately 708 km and a travel time estimate (converted into minutes in our data model).

The tool returns a structured RouteEstimate object to the agent containing: origin_query, destination_query, origin_lat, origin_lon, destination_lat, destination_lon, distance_km, duration_minutes, and raw_mode.

3.5 Weather Tool (Current Conditions)

The weather tool queries a weather API to obtain current conditions for the resolved origin and destination coordinates, returning a structured WeatherSnapshot. In addition to raw numerical values (temperature, precipitation, wind speed), the tool computes lightweight boolean indicators such as is_heavy_precipitation and is_strong_wind, which simplify downstream decision logic in the LLM prompt.

3.6 PydanticAI Framework Integration

PydanticAI manages the agent lifecycle and tool integration. Tool functions are declared as standard Python functions with type-annotated arguments. PydanticAI automatically generates OpenAI function call schema from Python type hints, passes tool descriptions to the LLM in the system prompt, validates tool invocation arguments against Pydantic models, and executes functions and returns results to the agent.

The agent is initialized with a system prompt encoding domain knowledge (country, task at hand) and decision rules. Conversation history is maintained across turns, enabling multi-turn refinement without re-invoking tools unnecessarily.

4. Agent Decision Logic

4.1 Transport Mode Selection Rules

The agent is instructed via a system prompt to choose a transport method by combining animal attributes inferred from the image caption and both route and weather information obtained through tool calls. The LLM acts as an orchestration and decision layer: it selects an appropriate routing profile, requests route estimates and destination weather, and produces a structured recommendation with qualitative safety considerations.

Tool-driven decision flow

For transport-related queries, the agent always performs two tool calls in a fixed order. First, it requests a route estimate between the user-provided origin and destination, selecting transport_hint $\in$ {car, truck, pedestrian, bicycle} based on the inferred animal size and location context. Second, it requests weather for the destination coordinates returned by the routing tool and uses this information to adjust the recommendation.

Selecting transport_hint for routing

The system prompt defines the following heuristics for choosing the routing profile:

- **Pedestrian:** If the locations appear extremely close (same building/block, neighboring entrances) and the animal is small/medium, the agent uses transport_hint="pedestrian" to estimate a walking-based time.
- **Bicycle:** If the distance is expected to be short (roughly a few kilometers), the animal is small (e.g., rodent, small cat, small dog), and the temperature is expected to be above −5 °C, the agent uses transport_hint="bicycle".
- **Truck:** If the animal is very large or likely requires bulky equipment (e.g., horse/cow, heavy crate), the agent uses transport_hint="truck".
- **Car:** Otherwise, the agent defaults to transport_hint="car".

Final transport recommendation rules

After both tools return results, the agent selects the transport method using the following criteria:

- **Pedestrian / hand carry:** Recommended only for very short distances (approximately up to 500 m), when the animal can be safely carried and brief outdoor exposure is not dangerous.
- **Bicycle:** Recommended for short trips (approximately 1–5 km) only if the animal is small and secured in a carrier, and conditions are not hazardous (temperature above −5 °C; no strong wind; no heavy precipitation; no obvious ice risk).
- **Car:** Default recommendation for most cases where the animal can fit comfortably in a car, and especially when the trip is too long for walking/cycling or when weather makes "light" options risky.
- **Truck:** Recommended only when a normal car is not suitable due to animal size/weight or special loading requirements.

Air travel as optional alternative

The agent does not treat air travel as a default mode. Instead, it may mention a plane only when both of the following conditions are satisfied:

1. The estimated driving time from the routing tool exceeds approximately 12 hours, and
2. The animal is plausibly acceptable under typical airline constraints (usually small/medium pets, no obvious extreme health issues).

When these conditions hold, the agent provides a rough plane-trip time estimate (e.g., "a few hours for flight plus check-in/transfers") and explicitly labels it as an approximation rather than a real-time schedule.

4.2 Design Choices and Constraints

Why separate captioning instead of end-to-end VLM? Qwen 2.5-VL integrates vision directly but sacrifices function calling reliability. Since our workflow depends on precise tool invocations (geocoding, routing, weather), we prioritize the text-only Instruct model's tool-use capability.

Why rule-based decision logic instead of pure LLM reasoning? A 7B model (especially one that's outdated) has limited world knowledge about animal physiology and transport

regulations. Encoding domain rules (temperature thresholds, distance categories) in the system prompt provides consistent, explainable decisions.

Why Yandex Geocoder + OpenRouteService instead of a single API? Yandex provides superior geocoding for Russian addresses (handles Cyrillic, local landmarks, abbreviations) but charges for routing. OpenRouteService offers free routing but weaker Russian geocoding. The hybrid approach maximizes quality while maintaining zero API costs.

Why OpenWeatherMap instead of Yandex.Weather? Yandex.Weather provides better forecasts for Russian cities but requires API key approval (multi-day delay). OpenWeatherMap offers instant free-tier access.

Why support multi-turn conversation? Real-world planning is iterative. Users may want to explore alternatives ("What if I go at night?", "What if it's a cat, not a dog?"). Maintaining conversation history allows the agent to answer follow-up questions without re-running expensive API calls.

5. Evaluation and Limitations

5.1 Qualitative Assessment

We tested the system on multiple scenarios covering different animal types (cat, dog, horse, hamster), distances (5 km to 1200 km), and weather conditions (cold, mild, hot). The agent correctly:

- Recommended walking/bicycle for small animals on short routes (<5 km) in mild weather
- Suggested personal car for medium distances (50–500 km) with appropriate rest stop reminders
- Flagged air travel for very long distances (>1000 km) or when weather posed risks
- Adjusted recommendations based on animal type (e.g., avoided bicycle for large dogs, recommended climate control for temperature-sensitive breeds)

Tool orchestration reliability: In all cases, the agent invoked tools in the correct sequence (routing → weather → recommendation). No instances of hallucinated tool calls or malformed JSON were observed.

5.2 Limitations

- **Image understanding:** Florence-2-base (0.23B parameters) is a lightweight model prioritizing speed over accuracy. It may lack understanding of rare breeds (e.g., Norwegian Cat).
- **Model limitations:** In certain cases, the agent displays lack of knowledge about certain breeds or is overly cautious. For example, it may recommend putting a jacket on a Siberian Husky at zero degrees Celsius.
- **Transport mode coverage:** The system recommends only ground transport modes (car, bicycle, courier). Air and rail options require additional APIs that are not implemented in this version.
- **Weather simplification:** Only current temperature is considered. A production system should integrate humidity (critical for reptiles), precipitation (affects walking/bicycle viability), multi-day forecasts, and route-level weather.

- **Regulatory compliance:** The agent does not verify documentation requirements (veterinary certificates, quarantine rules, permits for endangered species).
- **Distance estimation accuracy:** OpenRouteService provides geometric routing without real-time traffic. Yandex Routing API would improve estimates by incorporating current congestion.
- **Russian-only geocoding:** Yandex Geocoder is optimized for Russian addresses. International support would require a different geocoding service.

5.3 Comparison to Baselines

No directly comparable systems exist for multimodal animal transport planning with LLM agents. The closest precedent is the shelter transfer logistics tool by Wagner and Moore, which operates on batch transfers (multiple animals, multiple stops) versus our single animal, single-route focus, uses classical optimization versus LLM agent orchestration, and requires structured input versus our natural language + image interface.

6. Conclusion

This work demonstrates that a small open-source LLM (7B parameters) can effectively coordinate multiple external tools to solve a practical multi-step planning task. By separating vision (Florence-2 captioning) from reasoning (Qwen 2.5 agent), we achieve reliable function calling while maintaining multimodal input support. The system successfully integrates geospatial routing, weather data, and domain-specific decision rules to produce coherent transport recommendations from minimal user input (image + two addresses).

Key design insights:

1. Tool-use capability outweighs end-to-end integration for multi-step workflows with critical API dependencies
2. Rule-augmented LLM reasoning provides consistency when model scale limits world knowledge
3. Free API composition (Yandex Geocoder + OpenRouteService + OpenWeatherMap) enables zero-cost prototyping for niche applications
4. Hardware constraints (older GPUs) can be accommodated through careful model and framework version selection

Future work could extend the system with flight/rail APIs, cost estimation, veterinary compliance checking, and better LLM models or RAG systems to extend knowledge about species. Multi-objective optimization (time vs. cost vs. animal welfare) would transform the agent from a recommendation tool into a comprehensive transport planner. Nonetheless, the current implementation validates the core architecture: LLM agents can orchestrate heterogeneous APIs to deliver practical value, even under strict resource constraints.

References

[1] Yao, S., et al. (2022). ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*. https://arxiv.org/abs/2210.03629

[2] Qwen Team. (2024). Qwen 2.5: A Large-Scale Language Model. Alibaba Cloud. https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

[3] PydanticAI. (2024). Type-safe LLM Agents with Tool Support. https://docs.riza.io/guides/frameworks/pydantic-ai

[4] Dynamic Route Optimization in Last-Mile Delivery Using Predictive Analytics. (2024). European Centre for Research Training and Development-UK. https://eajournals.org/bjms/wp-content/uploads/sites/26/2024/10/Dynamic-Route-Optimization.pdf

[5] Wagner, T., & Moore, A. (2024). An Animal Transfer Logistics Support Tool. Pacific Transportation Research Center, USDOT. https://wpcdn.web.wsu.edu/cahnrs/uploads/sites/5/An-Animal-Transfer-Logistics-Support-Tool.pdf

[6] Dussert, G., et al. (2025). Zero-shot animal behaviour classification with vision language foundational models. *Methods in Ecology and Evolution*. https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.70059

[7] Microsoft. (2024). Florence-2: Advancing a Unified Representation for Visual Tasks. Hugging Face Model Hub. https://huggingface.co/microsoft/Florence-2-base

[8] Yandex LLC. Yandex Maps API: Geocoding and Routing Services. https://yandex.com/maps-api/

[9] OpenWeatherMap. Current Weather Data API Documentation. https://openweathermap.org/