# Road Condition Analysis Using Deep Learning


Topic: Road Condition Analysis Using Deep Learning

Team Name: Route 0x78

Team Number: 16

Team Members: Boldbaatar Amartuvshin, Yerbol Janbubu, Jamiyanpurev Ochir

## Table of Contents

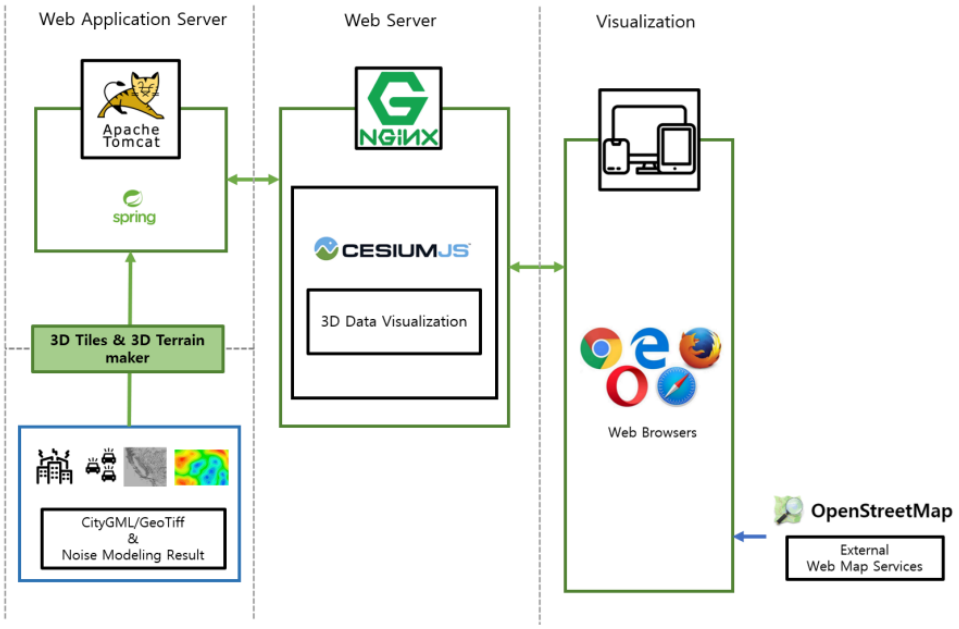



Figure 1: Software Configuration

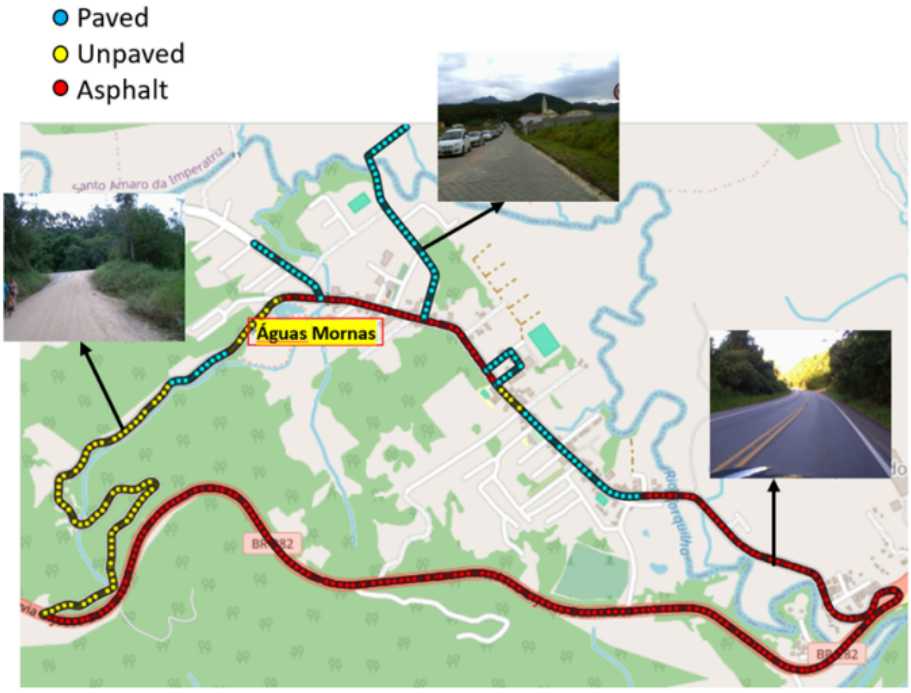



Figure 2: Visualization of Road Type Classification Results

# 1 Task Background and Objectives

## 1.1 Task Background

Urban Digital Twins (UDTs) enable cities to simulate and monitor real-world conditions in a virtual environment. Accurate road condition data is crucial for transportation planning, safety monitoring, and autonomous navigation. This project implements a Geo-AI module to classify road surface types (asphalt, paved, unpaved) using smartphone-collected imagery and geolocation data. This work contributes to an interoperable UDT ecosystem, as demonstrated in the OGC Urban Digital Twin Interoperability Pilot (UDTIP).

## 1.2 Dataset

- **Primary Dataset:** UN VMC dataset with geotagged images labeled by road type.
- **Supplementary Dataset:** RTK (Road Traversing Knowledge) dataset for model pre-training.
- **Output Format:** GeoPose and TrainingDML-AI compliant JSON.

## 1.3 Task Objectives

- Develop a smartphone-based data collection pipeline.
- Train a deep learning model to classify road surfaces.
- Build a Cesium-based visualization of classification results.
- Comply with OGC interoperability standards.

# 2 Requirements Analysis and Realistic Constraints

## 2.1 Software Requirements

- Python 3, PyTorch, OpenCV, FFmpeg
- CesiumJS, HTML/CSS/JS
- Flask for API development
- Label Studio for annotation
- Docker and GitHub for deployment/version control

## 2.2 Data Requirements

- Georeferenced road imagery sampled at 1Hz
- IMU and GPS metadata formatted into GeoPose
- Labeled training data for machine learning

### 2.3 Performance Requirements

- Model accuracy of at least 85%
- Inference capability in real-time or batch mode
- API response time below 2 seconds

### 2.4 Constraints

- Limited labeled data in some terrain types
- Varying lighting and image quality conditions
- Generalization across environments and devices
- Lightweight deployment requirements

## 3 Development Plan

### 3.1 Preprocessing

- Image and GPS synchronization
- Metadata conversion to GeoPose
- Sampling images at 1Hz and formatting as TrainingDML-AI

### 3.2 Annotation & Feature Preparation

- Manual labeling using Label Studio
- Class labels: Asphalt, Paved, Unpaved
- COCO-to-TDML format conversion

### 3.3 Model Training and Evaluation

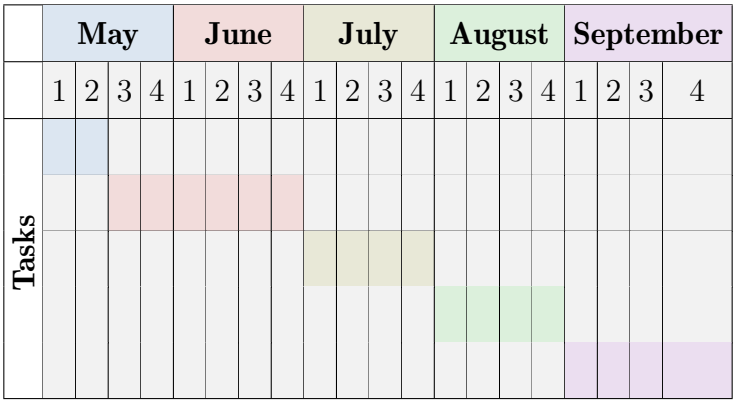
- Train classifiers (ResNet50, YOLOv5)
- Evaluate with precision, recall, and F1-score
- Optimize model and inference speed

### 3.4 Inference API & Visualization

- Flask-based inference API
- Map results to GeoPose coordinates
- Visualization using CesiumJS with interactive filters

# 4 Development Schedule and Role Division

## 4.1 Schedule




| Month | Tasks |
|---|---|
| May | Tool setup, data format study, hardware preparation |
| June | Data collection, preprocessing, image annotation |
| July | Model training, basic API setup |
| August | Model fine-tuning, inference integration |
| September | Visualization and UI development, full pipeline test |
| October | Final testing, documentation, and presentation |

## 4.2 Roles

- **Amartuvshin:** Model training, performance evaluation, API backend
- **Janbubu:** Visualization with Cesium & OSM, UI, OGC compliance
- **Jamiyanpurev:** In-vehicle image and GPS collection, metadata formatting

# 5 References


- OGC UDTIP `https://www.ogc.org/initiatives/ogc-urban-digital-twin-interoperability/`
- RTK Dataset — `https://www.kaggle.com/datasets/mitanshuchakrawarty/rtk-dataset`
- Label Studio — `https://labelstud.io`
- GeoPose Standard — `https://www.opengis.net/doc/IS/GeoPose/1.0`
- TrainingDML Standard — `https://daffodil.apache.org/tdml/`