SMART INDIA HACKATHON 2024

SMART INDIA HACKATHON 2024

TITLE PAGE

- **Problem Statement ID -** 1735
- **Problem Statement Title-** On-device semantic segmentation of WMS services with geospatial data export.
- **Theme-** Smart Automation
- **PS Category-** Software
- **Team ID-** Yet to be provided
- **Team Name -** Passengers

Passengers

SMART INDIA HACKATHON 2024

Necessity of the Problem and Why We Are Addressing It:

Challenges in Remote Sensing:

- Remote sensing applications, such as digitization and segmentation, are widely used in environmental monitoring, urban planning, and more.
- Existing segmentation solutions heavily rely on server-side GPUs, creating bottlenecks for non-technical users and reducing efficiency due to the reliance on internet connectivity and server availability.

Current Gaps:

- Server dependence increases latency, costs, and often leads to limited accessibility.
- Most segmentation tools are either too specialized or inaccessible to the broader community.

Why This Project?:

- There's a growing need for lightweight, user-friendly applications that can perform resource-heavy tasks locally, utilizing modern device GPUs or NPUs.
- By allowing on-device computation, users can achieve faster results, lower costs, and improve scalability.

Passengers

SMART INDIA HACKATHON 2024

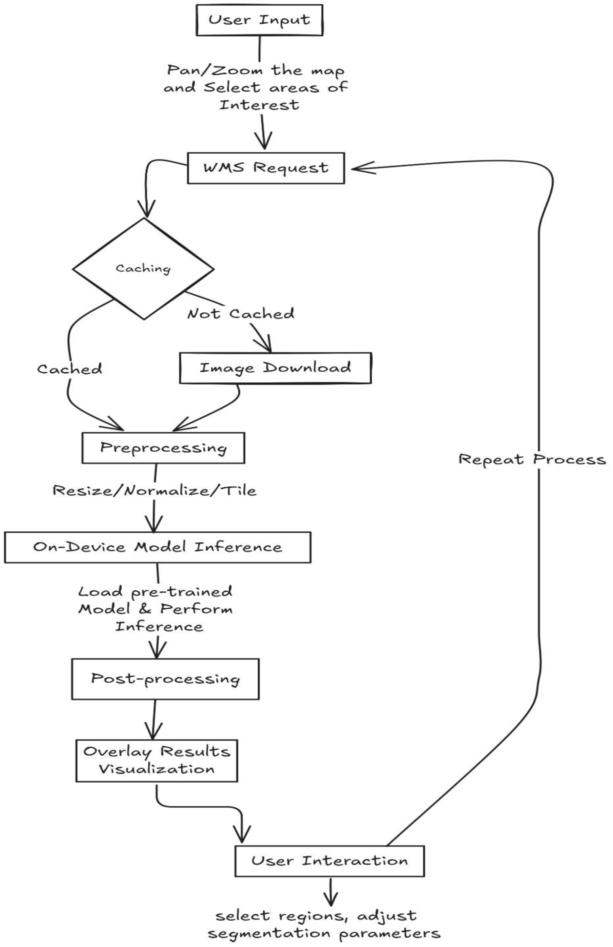
Proposed Solution of Prototype:

- ## IDEA:
  1. **Input Image via WMS**: The user selects a region of interest (ROI) using a Web Map Service (WMS), which serves geospatial imagery.
  2. **On-Device Segmentation**: The application performs semantic segmentation on the selected image using the device's GPU/NPU (hardware acceleration).
  3. **User Interaction**: The user can refine the segmentation by selecting, modifying, or adjusting the segmented features.
  4. **Geospatial Export**: Segmented features are exported in geospatial formats like GeoJSON or KML. This allows for further use in GIS software or WebGIS applications for analysis or visualization.

- ## SOLUTION ARCHITECTURE:
  - **Input**: Images are loaded via the WMS service based on the user's selected ROI.
  - **Processing**: On-device GPU/NPU handles segmentation using deep learning models such as U-Net or DeepLab. These models perform semantic segmentation in real-time, utilizing the device's hardware.
  - **Output**: Segmented features exported in geospatial format for further use in GIS software or WebGIS applications.

Why On-device GPU/NPU Matters: Emphasize the significance of moving away from server-based processing, and relying on local devices for cost-effective, faster, and more scalable solutions.

Passengers



SMART INDIA HACKATHON 2024

Passengers

Proposed Solution of Prototype contd:

- **How it Addresses the Problem**:
  - **Local Computation**: On-device GPU/NPU utilization ensures that the system operates without requiring powerful remote servers.
  - **User-Centric Design**: The interface is built for non-technical users, ensuring accessibility.
  - **Flexibility**: Works across multiple platforms (mobile, desktop, web), increasing its usability in various scenarios.

- **Innovation**:
  - **Interactive Segmentation**: A real-time, interactive segmentation interface that makes digitization faster and more precise.
  - **WMS Service Integration**: Full integration with OGC-compliant WMS services ensures the tool can be used with a wide variety of geospatial data providers.
  - **Geospatial Export Capabilities**: Efficient export to standard geospatial formats allows for seamless integration into existing workflows.

TECHNICAL APPROACH

Framework of Prototype:

High-level Architecture:

1. **Frontend**: Web/mobile app with a simple, interactive GUI for loading WMS layers and performing segmentation.
2. **Backend**: Local segmentation engine that runs on-device, utilizing frameworks such as TensorFlow Lite or PyTorch Mobile to execute models on GPU/NPU.
3. **WMS Integration**: Use libraries like `OWSLib` in Python or QGIS's Python API for loading WMS tiles.
4. **Export Module**: For converting segmented areas into GeoJSON/KML and saving or sending to external GIS systems.

Libraries/Technologies:

- **Frontend**: React (web), QGIS or ESRI Plugin (desktop).
- **Backend**: TensorFlow Lite, PyTorch Mobile, CUDA, or Vulkan API for GPU acceleration along with flask.
- **Data Handling**: GeoPandas, Shapely for geospatial data manipulation.

Image source: from respective company's website

Passengers

TECHNICAL APPROACH

SMART INDIA HACKATHON 2024

Development of Prototype:

Step 1: WMS Integration:

- **Details**: Utilize WMS capabilities to load raster images (satellite imagery, topographical maps, etc.) from OGC-compliant services like OpenStreetMap, SentinelHub, ESRI etc.
- **Backend**: Set up requests to OGC-compliant WMS services, fetching image tiles based on user-defined regions.
- **Frontend**: Display fetched WMS layers on a map interface (using
- Leaflet or OpenLayers) where users can choose regions.

Step 2: Semantic Segmentation on-device:

- **Model Selection**: Use pre-trained models such as U-Net or DeepLab optimized for mobile devices.
- **Model Conversion**: Convert the model using TensorFlow Lite or ONNX for compatibility with mobile devices.
- **Integration with GPU/NPU**: Use TensorFlow Lite's GPU delegate or PyTorch Mobile's Vulkan backend to run the model on the GPU.

Code Example:

python

```python
import geopandas as gpd
from shapely.geometry import Polygon
polygons = [Polygon(coords) for coords in segmented_regions]
geo_df = gpd.GeoDataFrame({'geometry': polygons})
geo_df.to_file('output.geojson', driver='GeoJSON')
```

python

```python
import tensorflow as tf
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

ode

```python
import geopandas as gpd
from shapely.geometry import Polygon
polygons = [Polygon(coords) for coords in segmented_regions]
geo_df = gpd.GeoDataFrame({'geometry': polygons})
geo_df.to_file('output.geojson', driver='GeoJSON')
```

code source: trial run for different parts

Passengers

TECHNICAL APPROACH

SMART INDIA HACKATHON 2024

Development of Prototype:

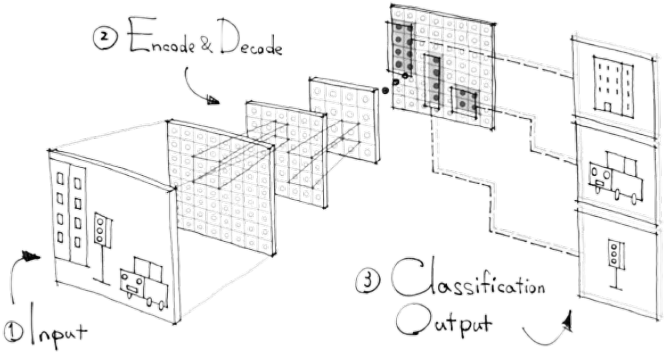


Image source:
https://medium.com/yodayoda/segmentation-for-creating-maps-92b8d926cf7e

Step 3: Interactive Segmentation Refinement:

- **Frontend**: Implement an interface for users to adjust the segmentation output, allowing for manual correction of features.
- **Backend**: Develop algorithms for user-driven corrections, ensuring that the changes can be applied seamlessly in real-time.

Step 4: Geospatial Data Export:

- **Conversion to GeoJSON/KML**: Post-process segmented regions into geospatial formats using libraries like GeoPandas and export as GeoJSON or KML.

Step 5: Performance Optimization:

- **Device-Specific Optimization**: Ensure efficient GPU/NPU utilization by profiling models on different device architectures (mobile vs desktop).

Passengers

TECHNICAL APPROACH

SMART INDIA HACKATHON 2024

```python
model = torch.hub.load('pytorch/vision:v0.10.0', 'deeplabv3_resnet50', pretrained=True)
model.eval()

# Function to preprocess the image for segmentation
def preprocess(image):
    transform = T.Compose([T.Resize(520), T.ToTensor()])
    return transform(image).unsqueeze(0)

# Function to segment the image using the pre-trained model
def segment_image(image_path):
    image = Image.open(image_path)
    input_image = preprocess(image)
    with torch.no_grad():
        output = model(input_image)['out'][0]
    output_predictions = output.argmax(0).numpy()  # Convert output to numpy array
    return output_predictions

# Function to convert segmentation output into polygons
def segment_to_polygons(segmented_image):
    # This is a placeholder function to convert segmented image pixels into polygons.
    # For simplicity, we'll create dummy polygons.
    polygons = [Polygon([(0, 0), (1, 0), (1, 1), (0, 1)])]  # Example polygon
    return polygons

# Endpoint to upload image and perform segmentation
@app.route('/segment', methods=['POST'])
def segment():
    if 'image' not in request.files:
        return jsonify({"error": "No image file provided"}), 400

    image = request.files['image']
    image_path = os.path.join("uploads", image.filename)
    image.save(image_path)

    # Run the segmentation function
    segmented_image = segment_image(image_path)
```

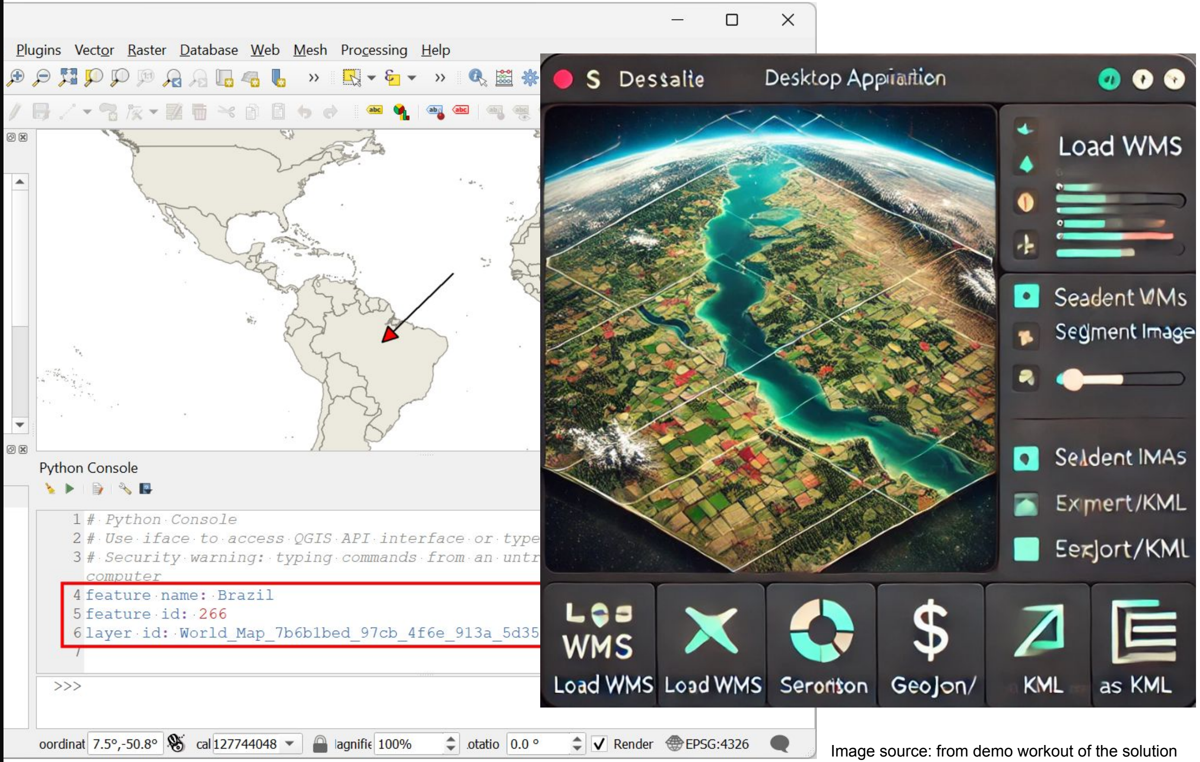


Image source: from demo workout of the solution

Passengers

KEY FEATURES

SMART INDIA HACKATHON 2024

Technical Feasibility: Modern mobile and desktop devices have adequate GPU/NPU support, and libraries like TensorFlow Lite, PyTorch Mobile, and Vulkan API facilitate efficient processing. The integration of WMS services with segmentation is achievable using existing APIs.

User Accessibility: The design aims for simplicity, making it accessible to both GIS experts and general users, particularly through a mobile or desktop QGIS plugin.

Cost and Efficiency: On-device processing eliminates the need for costly server infrastructure and allows real-time operation in the field without internet reliance.

Market Viability: The solution can be integrated into existing WebGIS platforms and used for various remote sensing applications, providing a unique selling point of offline capability.

- **On-device Processing**: Utilize the computational capabilities of GPUs/NPUs in modern devices.
- **User-friendly Interface**: Develop a simple, intuitive interface so non-technical users can segment images without needing complex technical knowledge.
- **Real-time Interaction**: Provide interactive segmentation, enabling users to visualize results on-screen in real-time.
- **Geospatial Export**: Support exporting segmented features in common geospatial formats such as GeoJSON and KML for integration with GIS tools.

Passengers

FEASIBILITY AND VIABILITY

SMART INDIA HACKATHON 2024

a) Challenges:

- **Device Compatibility**: Ensuring that the tool runs efficiently across a variety of devices with different GPU/NPU architectures.
- **Model Size and Speed**: Balancing model size with speed and accuracy to ensure real-time performance without consuming too much memory.
- **User Interface Design**: Making the tool accessible to non-technical users while still providing the necessary functionality.

b) Strategies:

- **Optimization Techniques**: Use model quantization and hardware-specific optimizations to reduce the size and increase the speed of models.
- **Incremental Deployment**: Start with desktop GPU-based solutions, then move to mobile implementations as optimizations are confirmed.
- **Extensive Testing**: Test the application on a wide range of devices to ensure consistent performance and compatibility.

Passengers

IMPACT AND BENEFITS

SMART INDIA HACKATHON 2024

Impacts and Benefits of solution:

1.Environmental Protection:

- **Early Detection and Response:** Rapid identification of oil spills allows for timely response and mitigation, minimizing the environmental damage caused by oil contamination.
- **Reduction in Ecosystem Damage:** Prompt action helps to protect marine life, shorelines, and sensitive ecosystems from prolonged exposure to oil pollutants.

2.Regulatory Compliance:

- **Enforcement of Regulations:** The ability to trace spills to specific vessels supports regulatory authorities in enforcing maritime pollution laws and holding responsible parties accountable.
- **Improved Reporting:** Provides evidence for accurate reporting and documentation of environmental incidents, aiding in compliance with international conventions and agreements.

Passengers

RESEARCH AND REFERENCES

SMART INDIA HACKATHON 2024

On-device Semantic Segmentation with GPUs/NPUs

- *Keypoint*: Utilizes device-local GPU/NPUs to process semantic segmentation, improving speed, reducing latency, and enabling offline functionality. **References**:
  - https://www.tensorflow.org/install/source_windows
  - https://pytorch.org/get-started/locally/

Integration with WMS Services (OGC-Compliant)

- *Keypoint*: Use OGC-compliant Web Map Service (WMS) to load geospatial data for segmentation, ensuring compatibility with standardized GIS systems. **References**:
  - https://www.ogc.org/standard/wms/
  - https://owslib.readthedocs.io/en/latest/

Geospatial Data Export (GeoJSON/KML)

- *Keypoint*: Export segmented regions as geospatial data (GeoJSON, KML) for integration with GIS tools, enabling end-to-end workflow from segmentation to geospatial analysis. **References**:
  - https://geopandas.org/en/stable/docs.html
  - https://shapely.readthedocs.io/en/stable/

Existing Technologies for Remote Sensing Applications

- *Keypoint*: Existing tools offer partial solutions (segmentation, WMS, export), but a comprehensive, GPU-accelerated on-device solution is lacking. **References**:
  - https://www.sentinel-hub.com/
  - https://arxiv.org/abs/1505.04597