Visual Spectrum Mine Detection

In this project, the latest advancements in AI and ML are leveraged to detect the PFM-1 mine in images captured through optical sensors operating in the visual spectrum. The dataset is generated with diverse backgrounds and environments to enhance the robustness of the algorithm.

Phases of the Algorithm

1. **Augmentation of True and False Photos**:
    - Original images are augmented using transformations to increase diversity and reduce overfitting during training.
2. **Conversion from `.png` or `.jpg` to `.tiff`**:
    - Embeds **(LAT, LON)** coordinates in the metadata (randomly generated around Paris).
3. **Training the YOLO Model**:
    - The YOLOv8 model is trained to detect the presence of the PFM-1 mine.
4. **Model Performance Verification**:
    - Test results are evaluated to extract:
        - **(LAT, LON)** of the detected mine: Converted from **(X, Y)** pixel coordinates.
        - Probability of mine presence.
        - Confidence level of detection.
5. **Visualization**:
    - Interactive map generation to display detected mine locations with probabilities and confidence levels.

Directory Structure

- `original/`: Raw images categorized into
    - `true/`: Images containing mines.
    - `false/`: Images without mines.
- `augmented/`: Augmented versions of images from `original/`
    - `true/` and `false/` subdirectories.
- `geo/`: GeoTIFF images with **(LAT, LON)** metadata
    - `true/` and `false/` subdirectories.
- `dataset/`: Prepared dataset for YOLO training
    - `train/`, `val/`, and `test/` directories for images and labels.
    - `results/`: Stores YOLO training outputs, including
        - **Weights**: Best-trained model weights.
        - Training batch visualizations and metrics.

Code Files

1. Notebook: `VSMDetection.ipynb`

The notebook serves as the central execution file, organizing and running the pipeline. It performs:

1. **Augmentation**: Calls the augmentation function to create diverse datasets.
2. **GeoTIFF Conversion**: Embeds geographic metadata.
3. **Training**: Trains YOLOv8 using the prepared dataset.
4. **Testing and Evaluation**: Evaluates model performance and saves results in `result.json`.
5. **Visualization**: Generates an interactive map of detected mines.

---

2. `augmentation.py`

- **Purpose**: Augments images to increase dataset diversity.
- **Key Functions**:
  - `process_images(src_folder, dst_folder)`: Applies transformations like flipping, rotating, brightness/contrast adjustments, and resizing.
  - **Output**: Augmented images saved to `augmented/`.

---

3. `geo_conversion.py`

- **Purpose**: Converts augmented `.png` or `.jpg` images into `.tiff` with embedded **(LAT, LON)** metadata.
- **Key Functions**:
  - `convert_images_to_tiff(src_folder, dest_folder)`: Assigns random coordinates near Paris and saves GeoTIFF files in `geo/`.

---

4. `model.py`

- **Purpose**: Handles dataset preparation, YOLO training, and evaluation.
- **Key Functions**:
  - `prepare_dataset(true_folder, false_folder, output_folder)`:
    - Splits images into `train`, `val`, and `test`.
    - Creates YOLO-format labels.
  - `train_model(model_name, epochs)`: Trains YOLOv8 with specified parameters.
  - `test_model(model_path, dataset_path)`: Runs inference on the test set.
  - `model_evaluation(results, dataset_path)`:
    - Converts bounding box coordinates from **(X, Y)** to **(LAT, LON)**.
    - Saves a JSON file with detailed test results.

---

5. `dataset.yaml`

- **Purpose**: Configuration file for YOLOv8 training.
- **Contents**:

```
train: <path-to-train-images>
val: <path-to-val-images>
nc: 1
names: ['mine']
```

- `nc`: Number of classes (1 for detecting mines).
- `names`: Class label.

Workflow

Step 1: Augmentation

- Run `augmentation.py` to process `original/true` and `original/false` images.
- Augmented images are saved in `augmented/`.

Step 2: GeoTIFF Conversion

- Use `geo_conversion.py` to embed random **(LAT, LON)** metadata.
- GeoTIFF images are saved in `geo/`.

Step 3: Dataset Preparation

- Execute `prepare_dataset` in `model.py` to split data into `train`, `val`, and `test`.
- YOLO-compatible datasets are saved in `dataset/`.

Step 4: Model Training

- Run `train_model` in `model.py` to train YOLOv8 with the prepared dataset.
- Outputs, including weights and visualizations, are saved in `results/`.

Step 5: Testing and Evaluation

- Run `test_model` and `model_evaluation` in `model.py`:
  - Generates bounding boxes.
  - Converts bounding box **(X, Y)** to **(LAT, LON)**.
  - Saves results in `result.json` with detection details.

Step 6: Visualization

- Use `visualize_detected_mines` in `evaluation.py`:
  - Generates `detected_mines_map.html` to visualize detected mines on a map of Paris.

Outputs

1. YOLO Training Results

- Saved in `results/run1/`:
  - **Model Weights**: Best model weights for inference.
  - **Metrics**: Training performance metrics (e.g., precision, recall).

2. Evaluation Results

- Saved as `result.json`:

- Detailed information for each test image, including:
  - `name`: Image name.
  - `mine_detected`: Yes/No.
  - `mine_present`: Yes/No (based on the dataset label).
  - `x, y`: Bounding box center (pixel coordinates).
  - `lat, lon`: Geographic coordinates.
  - `prob`: Probability of detection.
  - `conf`: Confidence level.

Example:


```
{
  "name": "IMG_0255_aug_5.tiff",
  "mine_detected": "yes",
  "mine_present": "yes",
  "x": 597.42041015625,
  "y": 808.9112548828125,
  "lat": 48.97964103030062,
  "lon": 2.011169568065193,
  "prob": 90.47,
  "conf": 0.90
}
```


3. Interactive Map

- Saved as `detected_mines_map.html`:
  - Visualizes detected mines on a map of Paris with interactive markers.
  - Each marker includes image name, probability, and confidence.

Screenshot:

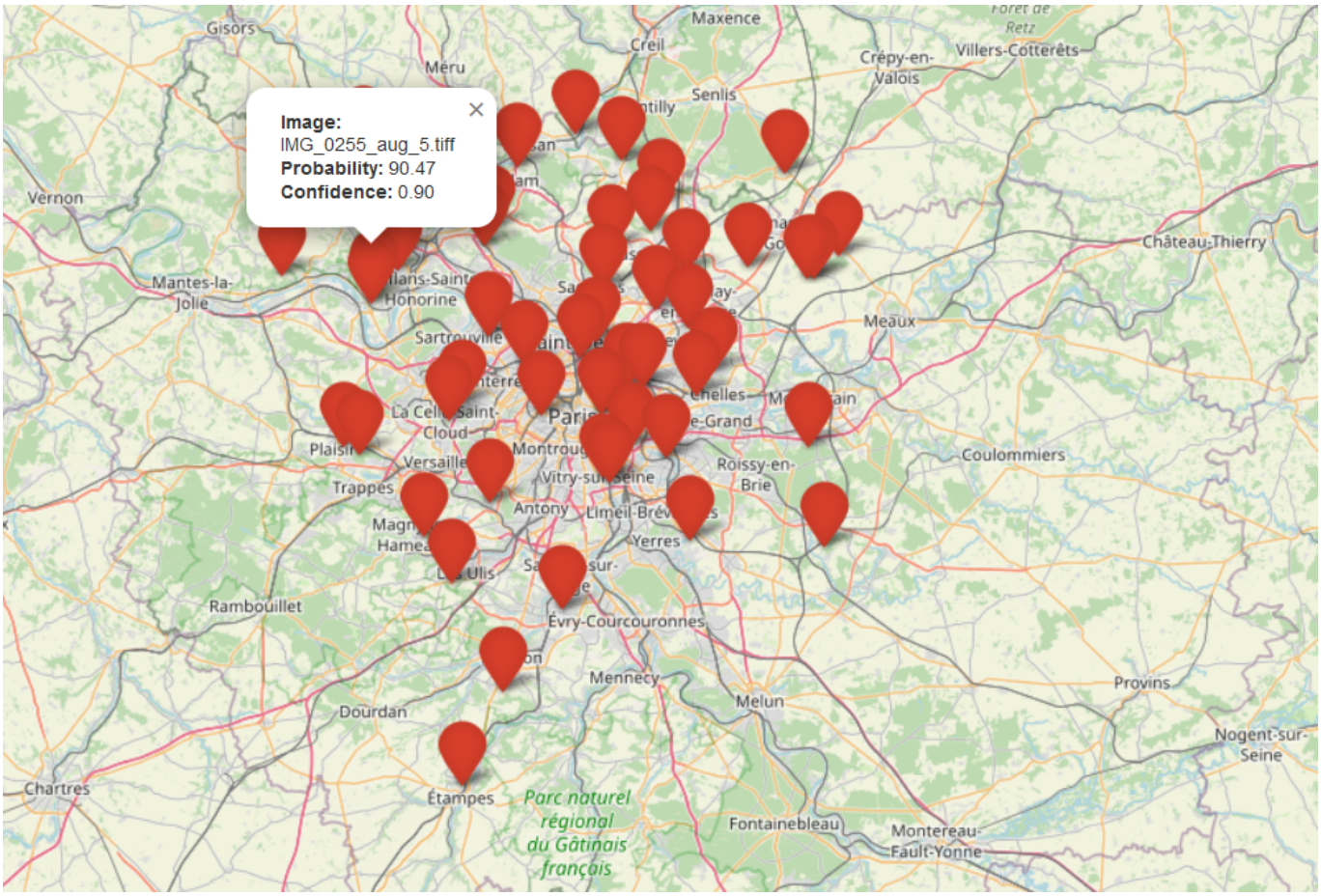
Image:
IMG_0255_aug_5.tiff
Probability: 90.47
Confidence: 0.90
Gisors
Maxence
Creil
Méru
Crépy-en-Valois
Villers-Cotterêts
Senlis
Vernon
Château-Thierry
Mantes-la-Jolie
Meaux
Sartrouville
Chelles
Coulommiers
Plaisir
Versailles
Montrouge
Vitry-sur-Seine
Roissy-en-Brie
Trappes
Antony
Limeil-Brévannes
Yerres
Rambouillet
Évry-Courcouronnes
Mennecy
Melun
Provins
Dourdan
Nogent-sur-Seine
Chartres
Étampes
Parc naturel régional du Gâtinais français
Fontainebleau
Montereau-Fault-Yonne